Universidade do Minho

Escola de Engenharia

Mestrado Integrado em Engenharia Informática

Unidade Curricular de

Bases de Dados

Ano Letivo de 2015/2016

OM

André Marinho, Bruno Ribeiro, Eduardo Cunha, Gabriel Fernandes

Novembro, 2015

Data de Receção	
Responsável	
Avaliação	
Observações	

OM

André Marinho, Bruno Ribeiro, Eduardo Cunha, Gabriel Fernandes

Novembro, 2015

Resumo

Este trabalho consiste no desenvolvimento de uma base de dados no âmbito da disciplina de Base de Dados, sendo que o tema é à nossa escolha sob a validação do professor.

Como tal, decidimos juntar o útil ao agradável e optámos por criar uma BD que agrupe informação relativa à música portuguesa. Este tema surgir devido à falta de informação relativa à música portuguesa, pois atualmente existe várias ferramentas que agrupar informação musical de todo o mundo, mas, no entanto, em relação à nossa não existe algo que a agrupe e divulgue.

Ao longo deste trabalho vamos apresentar todas as fases necessárias para a construção de uma base de dados, à qual nos baseamos no livro “Database Systems, A Practical Approach to Design, Implementation, and Management“ de Thomas Connolly e Carolyn Begg.

Área de Aplicação: Desenho e Arquitetura de uma Base de Dados para agrupar informação relativa à música portuguesa.

Palavras-Chave: Bases de Dados Relacionais, Modelo Conceptual, Modelo Lógico, Modelo Físico, Entidade, Atributo, Atributo Derivado, Relacionamento, Cardinalidade, Chave, Chave Candidata, Chave Primária, Chave Estrangeira, Sql, MySql, Mysql Workbench, brModelo.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1. Contextualização

Sendo Portugal um país com um riquíssimo e vasto background cultural, deparamo-nos com um obstáculo: todo esse leque cultural não se encontra devidamente organizado num só sítio. Em específico, no que toca à música feita por Portugueses ou por projetos criados em solo nacional, falhamos no que diz respeito à preservação desta informação.

Dada a tecnologia atual, era de se esperar algo que reunisse num só sítio toda a herança musical lusa. Até agora nenhum melómano se deu ao trabalho de conjugar a tecnologia atual com a sede de criar um repositório com todo esse conhecimento.

1.2. Apresentação do Caso de Estudo

Com o evoluir das tecnologias, foram criadas bases de dados que contêm informações relevantes sobre música. No entanto, a música portuguesa não acompanhou esta evolução, pelo que pouca informação sobre música portuguesa se encontra disponível. Para tal, e juntando o útil ao agradável, tomámos a iniciativa de construir uma base de dados para albergar esta informação em falta.

1.3. Motivação

As principais razões que nos motivaram a partir para este projeto foram: o gosto pessoal comum por música de todos os elementos deste grupo; o facto de não existir algo com uma grande quantidade de informação sobre música portuguesa; por fim, talvez um lado mais patriotista nos levou a optar por representar apenas a nossa música.

Considerando as razões acima representadas, decidimos então juntar o útil ao agradável e partir para o desenvolvimento deste projeto, que vai reunir as obras artísticas dos mais variados artistas portugueses, desde **A**va Inferi a **Z**eca Afonso.

1.4. Objetivos

Os principais objetivos que pretendemos alcançar na elaboração deste projeto são:

- Cimentar os conteúdos em causa na realização deste projeto, nomeadamente a metodologia usada para o desenvolvimento de um sistema de base de dados;
- Construir eficientemente uma base de dados que contenha informação referente às obras dos artistas portugueses;
- Utilizar futuramente este projeto como base para desenvolver uma base de dados ainda mais complexa que permita fazer um site, por exemplo.

1.5. Estrutura do Relatório

Inicialmente vamos apresentar os requisitos que temos de cumprir. A partir da análise dos mesmos, vamos apresentar todas as fases necessárias para o desenvolvimento de uma base de dados.

Numa primeira fase, vamos demonstrar o desenvolvimento do modelo conceptual deste projeto, procedendo à identificação das entidades, relacionamentos entre as mesmas, entre outros. No fim desta fase vamos apresentar o esquema conceptual referente a este projeto.

Já numa segunda fase, vamos traduzir o esquema conceptual para o esquema lógico, e proceder à sua validação, correspondendo esta fase à modelação lógica.

Numa última fase, vamos desenvolver o modelo físico do nosso projeto com o auxílio das duas fases anteriores, que consiste já no sistema de base de dados implementado, ou seja, pronto a utilizar.

Por fim, vamos concluir o nosso projeto com uma análise crítica do trabalho realizado, apresentando possíveis melhorias a aplicar futuramente.

2. Análise de Requisitos

A organização desta informação deverá ser estruturada da seguinte forma: uma banda é registada na base de dados com o seu nome, logotipo, país e cidade de origem, estado atual (ativa, acabada, em hiato, desconhecido), ano em que se formou, uma descrição e, por fim, informação sobre os temas líricos abordados na sua discografia; uma banda também se caracteriza por possui um ou mais géneros (que se caracterizam pelo nome do género, por exemplo, rock progressivo), sendo que estes se podem relacionar entre si formando subgéneros (por exemplo, rock progressivo é um subgénero de rock); uma banda possui também álbuns, que contém título, capa, data de lançamento, editora, duração e descrição, podendo ainda um álbum pertencer a bandas diferentes; um álbum possui músicas, que se caracterizam por possuir um titulo, duração e letra, podendo uma música fazer parte de álbuns diferentes; por fim, uma banda possui um ou mais membros, sendo estes caracterizados pelo seu nome, tendo em atenção que uma banda pode não ser constituída pelos mesmos membros ao longo da sua atividade.

Através do que foi demonstrado na apresentação do caso de estudo, verificámos que a base de dados deve:

- Permitir a inserção/remoção de bandas;
- Permitir a inserção/remoção de álbuns;
- Permitir a inserção/remoção de músicas;
- Pesquisar lista de bandas por nome;
- Pesquisar lista de bandas por género;
- Pesquisar lista de bandas por álbum;
- Pesquisar lista de bandas por música;
- Obter informações sobre bandas, álbuns e músicas;
- Entre outros.

3. Modelo Conceptual

3.1. Identificar entidades

Neste primeiro passo vamos analisar os requisitos apresentados no capítulo anterior, assim como identificar quais as entidades que vamos ter de representar no nosso modelo conceptual.

Posto isto, e depois da análise feita aos requisitos, identificámos cinco entidades, que passámos a enumerar: Banda, Membro, Álbum, Música e Género.

Entidade	Descrição	Ocorrência
Banda	Termo geral que descreve todas as bandas.	Cada banda portuguesa.
Música	Termo geral que descreve todas as músicas das bandas.	Cada música de cada banda, presente ou não nos seus álbuns.
Álbum	Termo geral que descreve todos os álbuns das bandas.	Cada álbum de cada banda.
Membro	Termo geral que descreve todos os membros das bandas.	Cada membro de cada banda.
Género	Termo geral que descreve todos os (sub)géneros das bandas.	Cada (sub)género existente na música portuguesa.

Tabela 1: Entidades.

3.2. Identificar relacionamentos entre entidades

Tendo já sido reconhecidas as entidades do projeto, vamos identificar que relações existem entre estas, bem como a sua cardinalidade e significado.

A partir da análise dos requisitos, conseguimos identificar cinco relacionamentos entre as entidades, que desde já passámos a demonstrar através de um diagrama ER:

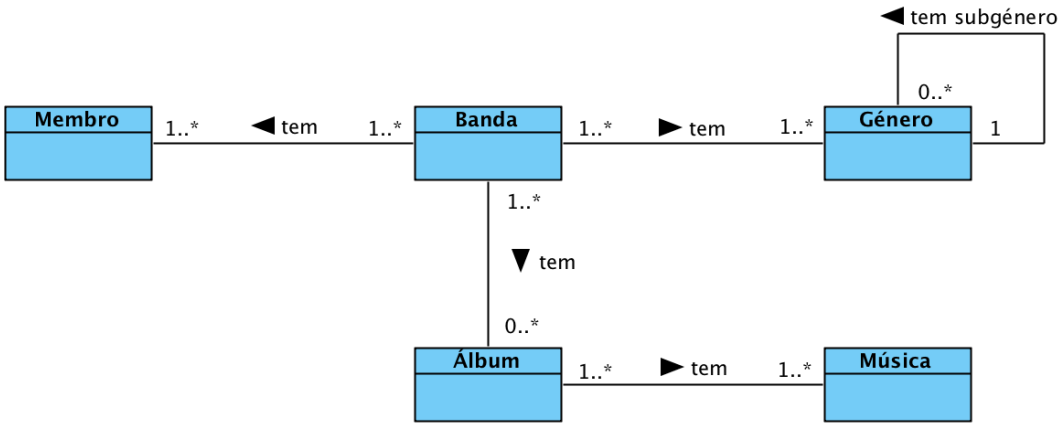

Figura 1: Diagrama ER de relacionamentos entre entidades.

Entidade	Multiplicidade	Relação	Multiplicidade	Entidade
Banda	1..N	Tem	1..N	Membro
Banda	1..N	Tem	0..N	Álbum
Banda	1..N	Tem	1..N	Género
Álbum	1..N	Tem	1..N	Música
Género	1..1	Tem Subgénero	0..N	Género

Tabela 2: Relacionamentos entre entidades.

3.3. Identificar e associar atributos com entidades ou relações

De seguida, a partir das entidades e já com as relações estabelecidas, é possível determinar os atributos de cada entidade e relação como descrito nas seguintes tabelas:

Entidade	Atributos	Descrição	Tipo e Tamanho	*Nulls*
Banda	idBanda	Identificador único de uma banda	Inteiro	Não
	nome	Nome da banda	100 caracteres variáveis	Não
	logotipo	Logotipo da banda	Imagem	Sim
	anoFormacao	Ano de formação da banda	Data	Não
	cidadeOrigem	Cidade de origem da banda	30 caracteres variáveis	Não
	paisOrigem	País de origem da banda	30 caracteres variáveis	Não
	estado	Estado atual da banda	20 caracteres variáveis	Não
	temasLiricos	Temas líricos abordados pela banda	Número indefinido de caracteres variáveis	Sim
	descricao	Descrição da banda	Número indefinido de caracteres variáveis	Sim
Membro	idMembro	Identificador único de um membro de uma banda	Inteiro	Não
	nome	Nome do membro de uma banda	100 caracteres variáveis	Não
Álbum	idAlbum	Identificar único de um álbum	Inteiro	Não
	titulo	Título do álbum	100 caracteres variáveis	Não
	capa	Capa do álbum	Imagem	Sim
	dataLancamento	Data de lançamento do álbum	Data	Não
	editora	Editora que gravou o álbum	50 caracteres variáveis	Sim
	descricao	Descrição do álbum	Número indefinido de caracteres variáveis	Sim
Música	idMusica	Identificador único de uma música	Inteiro	Não
	titulo	Título da música	50 caracteres variáveis	Não
	duracao	Duração da música	Tempo	Não
	letra	Letra da música	Número indefinido de caracteres variáveis	Sim

Género	idGenero	Identificador único de um género	Inteiro	Não
	nome	Nome do género musical	20 caracteres variáveis	Não

Tabela 3: Atributos de cada entidade.

Relação	Atributos	Descrição	Tipo e Tamanho	*Nulls*
Banda - Música	estado	Estado atual do membro em relação à banda (se ainda integra a banda, se é membro de sessão, entre outros)	20 caracteres variáveis	Não

Tabela 4: Atributos de cada relação.

3.4. Determinar o domínio dos atributos

Após a análise aos atributos referentes às entidades e relações, é possível identificar o domínio, tipo e formato dos mesmos como é demonstrado nas seguintes tabelas:

Entidade	Atributos	Domínio	Tipo e Tamanho
Banda	idBanda	Inteiro único maior que 0	INT
	nome	Nome da banda válido	VARCHAR(100)
	logotipo	Logotipo da banda válido	LONGBLOB
	anoFormacao	Ano de formação da banda válido	DATE
	cidadeOrigem	Cidade de origem da banda válida	VARCHAR(30)
	paisOrigem	País de origem da banda válido	VARCHAR(30)
	estado	Estado atual da banda válido	VARCHAR(20)
	temasLiricos	Texto com os temas líricos abordados pela banda	TEXT
	descricao	Texto descritivo da banda	TEXT
Membro	idMembro	Inteiro único maior que 0	INT
	nome	Nome do membro de uma banda válido	VARCHAR(100)
Álbum	idAlbum	Inteiro único maior que 0	INT
	titulo	Título do álbum válido	VARCHAR(100)
	capa	Capa do álbum válida	LONGBLOB
	dataLancamento	Data de lançamento do álbum válida	DATE
	editora	Editora que gravou o álbum válida	VARCHAR(50)
	descricao	Texto descritivo do álbum	TEXT
Música	idMusica	Inteiro único maior que 0	INT
	titulo	Título da música válido	VARCHAR(50)
	duracao	Duração da música válida	TIME
	letra	Texto com a letra da música	TEXT
Género	idGenero	Inteiro único maior que 0	INT
	nome	Nome do género musical válido	VARCHAR(20)

Tabela 5: Domínios dos atributos de cada entidade.

Relação	Atributos	Domínio	Tipo e Tamanho
Banda - Música	estado	Estado atual do membro válido	VARCHAR(20)

Tabela 6: Domínios dos atributos de cada relação.

3.5. Determinar chaves candidatas, primárias e alternativas

Na tabela em baixo está representada a chave primária de cada entidade e, se existirem, chaves alternativas. Através da análise dos atributos de cada entidade, apenas encontrámos uma chave alternativa, pois os outros atributos não conseguem representar univocamente cada tuplo.

Entidade	Chave Primária	Chave Alternativa
Banda	idBanda	
Membro	idMembro	
Álbum	idAlbum	
Música	idMusica	
Género	idGenero	Nome

Tabela 7: Chaves primárias e alternativas de cada entidade.

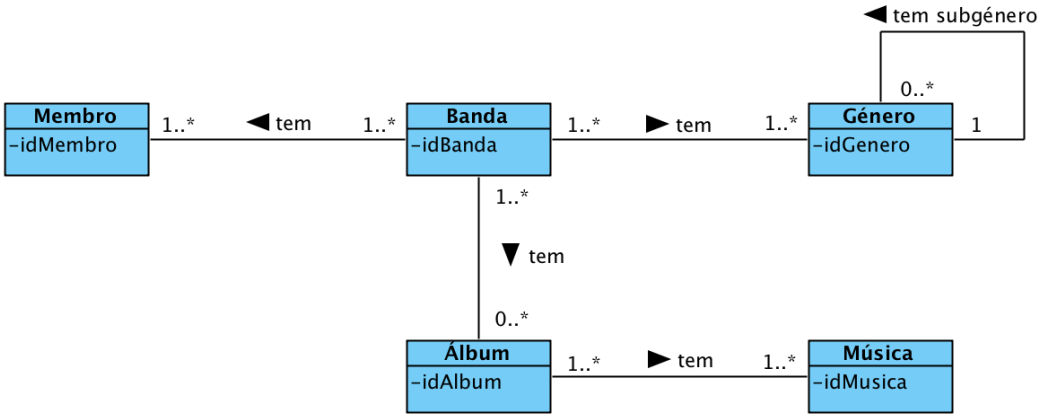

Figura 2: Diagrama ER de relacionamentos entre entidades com chaves primárias.

3.6. Verificar existência de redundâncias

Neste passo vamos examinar o modelo conceptual com o objetivo de verificar a existência de redundâncias, e em caso afirmativo, removê-las.

Para isso temos de:

- Voltar a verificar relacionamentos de 1:1;
- Remover relações redundantes;
- Verificar se modificações temporais de relações entre entidades podem causar redundâncias.

Em relação ao primeiro ponto, o nosso projeto não possui relacionamentos de 1:1, pelo que neste ponto não haverá redundâncias; já em relação ao segundo ponto, não encontrámos relações redundantes entre entidades; por fim, e em relação ao último ponto, verificámos que as relações que temos são estritamente necessárias, e que modificações temporais de relações

entre entidades (um membro de uma banda deixar esta, por exemplo) não resultam em redundância.

Em suma, não foi detetado nenhuma redundância no modelo.

3.7. Validar o modelo conceptual sobre transações

O modelo, para preencher os requisitos, deve cumprir as seguintes *queries*:

a) Dado uma música, determinar a que álbuns pertence;
b) Dado uma música, determinar a que banda(s) pertence;
c) Dado um álbum, determinar que músicas este contém;
d) Dado um álbum, determinar a que banda(s) pertence;
e) Dado uma banda, determinar a lista de músicas da mesma;
f) Dado uma banda, determinar a lista de álbuns da mesma;
g) Dado uma banda, determinar a lista de géneros da mesma;
h) Dado uma banda, determinar a lista de membros da mesma;
i) Dado um membro, determinar em que banda(s) toca;
j) Dado um género, determinar a lista de bandas do mesmo.

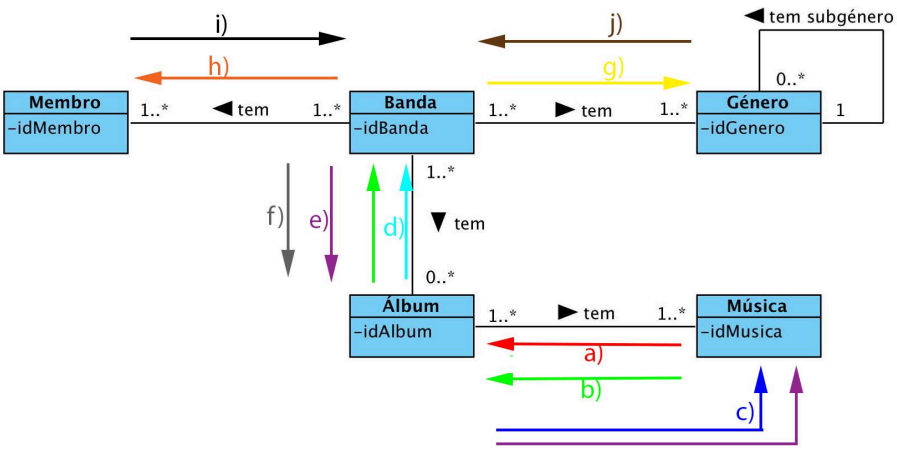

Figura 3: Mapa de transações do modelo conceptual.

Pela observação do diagrama acima representado, verificámos que o modelo cumpre as transações estipuladas, concluindo então que o modelo se encontra válido até este ponto.

3.8. Esquema conceptual e revisão

Tendo em consideração todos os passos realizados anteriormente, estamos aptos a representar o esquema conceptual do nosso projeto, que se apresenta de seguida:

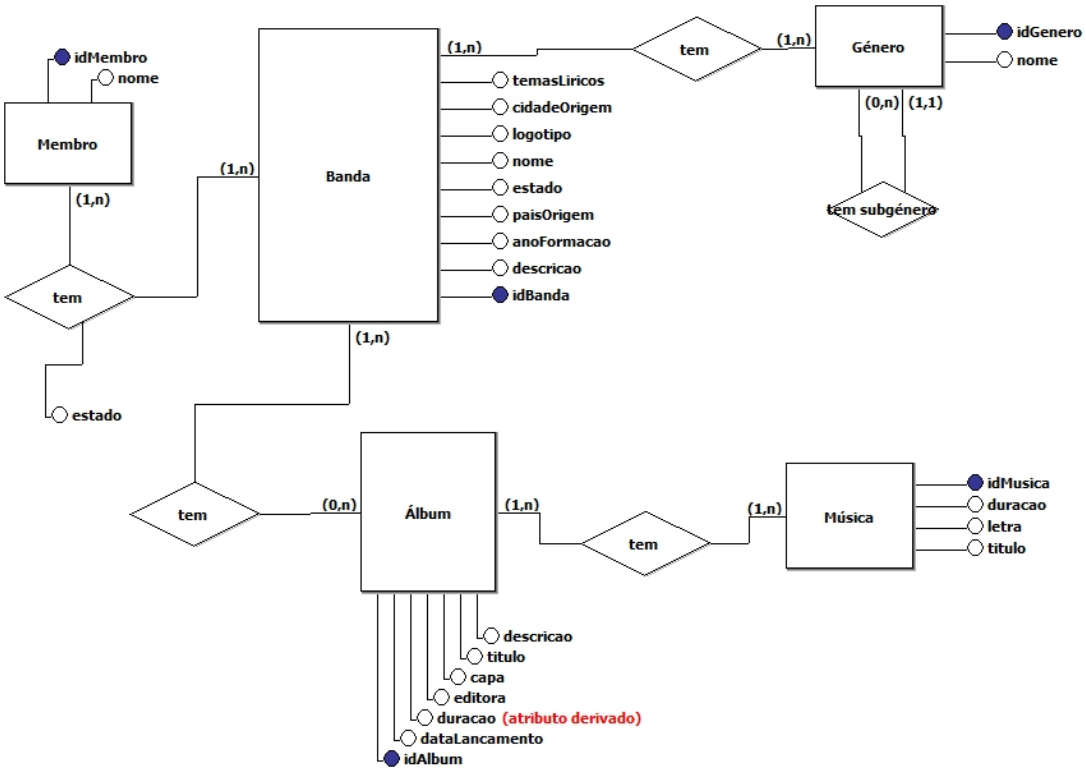

Figura 4: Esquema conceptual do projeto desenvolvido no brModelo.

Por fim, observando o nosso modelo, podemos afirmar que não encontrámos nenhuma anomalia no mesmo e que este cumpre as transações necessárias definidas nos requisitos, concluindo assim com sucesso a fase de modulação conceptual do nosso projeto.

4. Modelo Lógico

4.1. Passagem do modelo conceptual para o modelo lógico

Tendo como base o esquema conceptual desenvolvido no capítulo anterior, vamos então contruir o esquema lógico correspondente, que se apresenta de seguida:

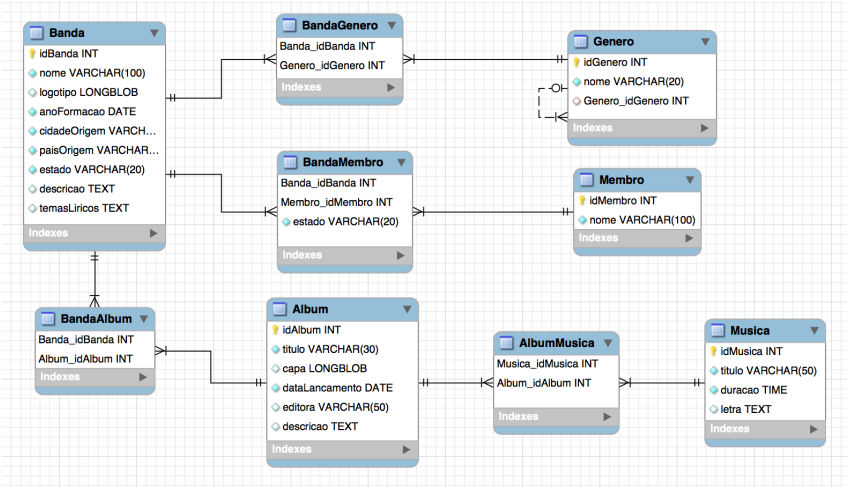

Figura 5: Esquema lógico derivado do esquema conceptual.

4.2. Validar modelo lógico através da normalização

O modelo lógico já se encontra na primeira forma normal. Convém salientar que na primeira fase do trabalho tínhamos um atributo multivalor (“membros”) na entidade “Banda”. Caso tivéssemos continuando com o modelo conceptual inalterado da primeira fase, teríamos obviamente violado a primeira forma normal, de maneira a que teríamos de criar uma nova entidade para a primeira forma normal ser válida. No entanto, decidimos reestruturar o nosso modelo conceptual considerando já “Membro” como entidade.

De modo a respeitar a segunda e a terceira forma normal, tivemos que alterar o modo como representamos a origem da “Banda”. Inicialmente tínhamos tanto a “Cidade” como o “Pais” como atributo da entidade “Banda”. Na realidade, sabemos que existe uma relação de implicação entre uma cidade e um país. Por exemplo, dada uma certa cidade, facilmente conseguimos saber a que país pertence. De acordo com a terceira forma normal não faz sentido estarmos a repetir a informação referente à cidade e ao país por cada tuplo, portanto, tanto para a “Cidade” como para o “Pais”, criámos uma nova tabela. Também fizemos o mesmo para o “estado” da “Banda” e do “Membro”.

Posto isto, e já depois de identificadas as alterações necessárias para a validação do modelo lógico através da normalização, o esquema lógico encontra-se da seguinte forma:

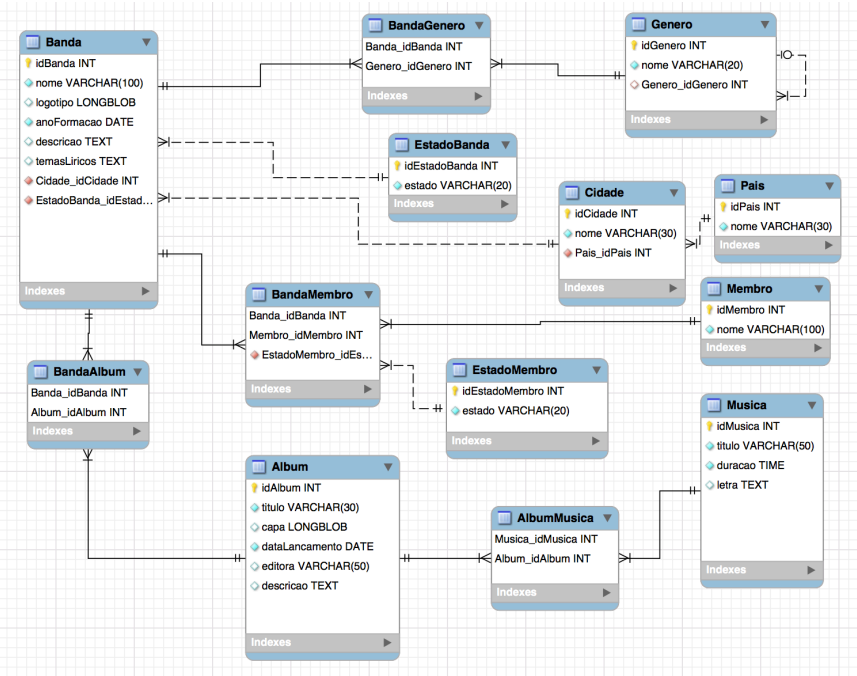

Figura 6: Esquema lógico depois de aplicada as regras de normalização.

4.3. Validar modelo lógico através das transações

O modelo lógico, para ser válido, deve cumprir as mesmas *queries* analisadas aquando da definição do modelo conceptual. Posto isto, dever cumprir o seguinte:

a) Dado uma música, determinar a que álbuns pertence;
b) Dado uma música, determinar a que banda(s) pertence;
c) Dado um álbum, determinar que músicas este contém;
d) Dado um álbum, determinar a que banda(s) pertence;
e) Dado uma banda, determinar a lista de músicas da mesma;
f) Dado uma banda, determinar a lista de álbuns da mesma;
g) Dado uma banda, determinar a lista de géneros da mesma;
h) Dado uma banda, determinar a lista de membros da mesma;
i) Dado um membro, determinar em que banda(s) toca;
j) Dado um género, determinar a lista de bandas do mesmo.

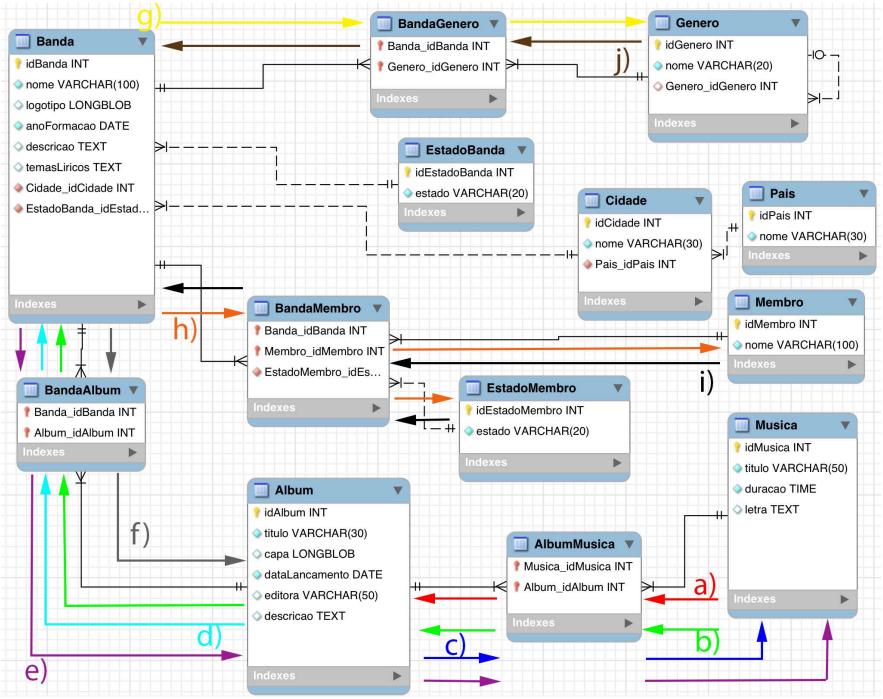

Figura 7: Mapa de transações do modelo lógico.

Através da análise da fig. 7, concluímos que o nosso modelo lógico cumpre todas as transações estipuladas, pelo que o modelo se encontra válido até este passo.

4.4. Verificar restrições de integridade

Para termos uma base de dados seja integra, é preciso verificar as restrições que existem no modelo, bem como se estas são válidas. De forma a garantirmos que a informação nos atributos seja válida, temos de garantir certas condicionantes. Por exemplo, uma banda não existe sem nome, assim como um álbum, uma música, uma cidade, etc. Isto significa que este atributo não pode ser *null*. Da mesma forma, uma banda não existe sem tocar um género de música (pelo menos).

Também tivemos de verificar se os tipos dos atributos e os seus domínios estavam corretos, bem como a multiplicidade das relações. Nesta fase, tivemos a prova de que podemos estar a partir de um modelo anterior com erros, ou a partir de uma lista de requisitos incompleta, dando conta dos erros nesta fase de verificação.

Para além de certos atributos não poderem ser nulos, visto que na vida real também não o são, há um certo tipo de atributos que também não podem ser nulos, como o caso das chaves primárias.

De modo a verificar a integridade de referências, temos um caso na nossa base de dados em que a relação é opcional, isto é, a chave estrangeira pode ser nula. No caso do género, um género pode não ter “género-pai”, guardando nesse caso *null* na chave estrangeira.

4.5. Análise do crescimento futuro

Tal como dito na secção da motivação, decidimos que a base de dados ia suportar o mundo musical português. No entanto, para que posteriormente, caso seja desejado, a base de dados possa albergar outros mundos, decidimos acrescentar mais uma entidade chamada “Pais” que, como sugere, possa suportar mais países de onde as bandas possam ser oriundas.

Estamos assim a garantir que o modelo de base de dados atual seja extensível, podendo evoluir facilmente suportando, sem muitas alterações nos dados atuais, novas funcionalidades. Claro que, neste caso, poderíamos excluir do nosso modelo a entidade “Pais”, mas foi decidido inclui-la pelos motivos acima descritos.

Num cenário empresarial, numa dimensão muito maior que um projeto académico, é óbvio que é difícil saber o que futuramente poderá ser exigido, mas sabendo *à priori* qual o rumo em que o projeto pode evoluir, é mais rentável incluirmos esse suporte, mesmo que numa dimensão mais pequena.

4.6. Revisão do modelo lógico

Sendo este o último passo para a modelação lógica do projeto, não deixa de ser o mais importante, pois é neste passo que se verifica tudo o que se fez até ao momento e se procura encontrar erros, ou até acrescentar transações que se consideram necessárias.

Depois de analisado tudo o que se fez para trás, verificámos que o modelo lógico cumpre os requisitos pré-definidos, e também que cumpre todas as transações necessárias estipuladas. Além disto, verificámos também que o modelo se encontra válido através das regras de normalização.

Por fim, e considerando a análise a este modelo, concluímos que a fase de modulação lógica do nosso projeto foi concluída com sucesso.

5. Modelo Físico

5.1. Tradução do modelo lógico para um SGBD

Para a tradução do modelo lógico para o modelo conceptual, usámos a ferramenta MySQL Workbench. Primeiramente, implementámos as tabelas, as relações e os atributos correspondente a cada. Depois de termos todo o modelo lógico traduzido para o SGBD, apenas precisámos de fazer "*Forward Enginner*" e este gera automaticamente o esquema físico do nosso projeto (na fig. 6 apresentámos o modelo lógico traduzido no MySQL Workbench).

5.2. Esquema físico

Depois de fazer o descrito na secção anterior, o SGBD criou um *script* de criação do nosso projeto, que passámos a mostrar:

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`Pais`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Pais` (
  `idPais` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(30) NOT NULL,
  PRIMARY KEY (`idPais`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Cidade`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Cidade` (
  `idCidade` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(30) NOT NULL,
  `Pais_idPais` INT NOT NULL,
  PRIMARY KEY (`idCidade`),
```

```
  INDEX `fk_cidade_pais1_idx` (`Pais_idPais` ASC),
  CONSTRAINT `fk_cidade_pais1`
    FOREIGN KEY (`Pais_idPais`)
    REFERENCES `mydb`.`Pais` (`idPais`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`EstadoBanda`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`EstadoBanda` (
  `idEstadoBanda` INT NOT NULL AUTO_INCREMENT,
  `estado` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`idEstadoBanda`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Banda`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Banda` (
  `idBanda` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(100) NOT NULL,
  `logotipo` LONGBLOB NULL,
  `anoFormacao` DATE NOT NULL,
  `descricao` TEXT NULL,
  `temasLiricos` TEXT NULL,
  `Cidade_idCidade` INT NOT NULL,
  `EstadoBanda_idEstadoBanda` INT NOT NULL,
  PRIMARY KEY (`idBanda`),
  INDEX `fk_Banda_cidade1_idx` (`Cidade_idCidade` ASC),
  INDEX `fk_Banda_EstadoBanda1_idx` (`EstadoBanda_idEstadoBanda` ASC),
  CONSTRAINT `fk_Banda_cidade1`
    FOREIGN KEY (`Cidade_idCidade`)
    REFERENCES `mydb`.`Cidade` (`idCidade`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Banda_EstadoBanda1`
    FOREIGN KEY (`EstadoBanda_idEstadoBanda`)
    REFERENCES `mydb`.`EstadoBanda` (`idEstadoBanda`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Genero`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Genero` (
  `idGenero` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(20) NOT NULL,
  `Genero_idGenero` INT NULL,
  PRIMARY KEY (`idGenero`),
  INDEX `fk_Género_Género_idx` (`Genero_idGenero` ASC),
  CONSTRAINT `fk_Género_Género`
    FOREIGN KEY (`Genero_idGenero`)
    REFERENCES `mydb`.`Genero` (`idGenero`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Album`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Album` (
  `idAlbum` INT NOT NULL AUTO_INCREMENT,
  `titulo` VARCHAR(30) NOT NULL,
  `capa` LONGBLOB NULL,
  `dataLancamento` DATE NOT NULL,
  `editora` VARCHAR(50) NULL,
  `descricao` TEXT NULL,
  PRIMARY KEY (`idAlbum`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Musica`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Musica` (
  `idMusica` INT NOT NULL AUTO_INCREMENT,
  `titulo` VARCHAR(30) NOT NULL,
  `duracao` TIME NOT NULL,
  `letra` TEXT NULL,
  PRIMARY KEY (`idMusica`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`BandaGenero`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`BandaGenero` (
  `Banda_idBanda` INT NOT NULL,
  `Genero_idGenero` INT NOT NULL,
  INDEX `fk_BandaGenero_Banda1_idx` (`Banda_idBanda` ASC),
  INDEX `fk_BandaGenero_Genero1_idx` (`Genero_idGenero` ASC),
  PRIMARY KEY (`Banda_idBanda`, `Genero_idGenero`),
  CONSTRAINT `fk_BandaGenero_Banda1`
    FOREIGN KEY (`Banda_idBanda`)
    REFERENCES `mydb`.`Banda` (`idBanda`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_BandaGenero_Genero1`
    FOREIGN KEY (`Genero_idGenero`)
    REFERENCES `mydb`.`Genero` (`idGenero`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`BandaAlbum`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`BandaAlbum` (
  `Banda_idBanda` INT NOT NULL,
  `Album_idAlbum` INT NOT NULL,
  INDEX `fk_BandaAlbum_Banda1_idx` (`Banda_idBanda` ASC),
  INDEX `fk_BandaAlbum_Album1_idx` (`Album_idAlbum` ASC),
```

```
  PRIMARY KEY (`Banda_idBanda`, `Album_idAlbum`),
  CONSTRAINT `fk_BandaAlbum_Banda1`
    FOREIGN KEY (`Banda_idBanda`)
    REFERENCES `mydb`.`Banda` (`idBanda`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_BandaAlbum_Album1`
    FOREIGN KEY (`Album_idAlbum`)
    REFERENCES `mydb`.`Album` (`idAlbum`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`AlbumMusica`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`AlbumMusica` (
  `Musica_idMusica` INT NOT NULL,
  `Album_idAlbum` INT NOT NULL,
  INDEX `fk_AlbumMusica_Musica1_idx` (`Musica_idMusica` ASC),
  INDEX `fk_AlbumMusica_Album1_idx` (`Album_idAlbum` ASC),
  PRIMARY KEY (`Musica_idMusica`, `Album_idAlbum`),
  CONSTRAINT `fk_AlbumMusica_Musica1`
    FOREIGN KEY (`Musica_idMusica`)
    REFERENCES `mydb`.`Musica` (`idMusica`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_AlbumMusica_Album1`
    FOREIGN KEY (`Album_idAlbum`)
    REFERENCES `mydb`.`Album` (`idAlbum`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Membro`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Membro` (
  `idMembro` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`idMembro`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`EstadoMembro`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`EstadoMembro` (
  `idEstadoMembro` INT NOT NULL AUTO_INCREMENT,
  `estado` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`idEstadoMembro`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`BandaMembro`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`BandaMembro` (
```

```
  `Banda_idBanda` INT NOT NULL,
  `Membro_idMembro` INT NOT NULL,
  `EstadoMembro_idEstadoMembro` INT NOT NULL,
  INDEX `fk_BandaMembro_Banda1_idx` (`Banda_idBanda` ASC),
  INDEX `fk_BandaMembro_Membro1_idx` (`Membro_idMembro` ASC),
  PRIMARY KEY (`Banda_idBanda`, `Membro_idMembro`),
  INDEX `fk_BandaMembro_EstadoMembro1_idx` (`EstadoMembro_idEstadoMembro` ASC),
  CONSTRAINT `fk_BandaMembro_Banda1`
    FOREIGN KEY (`Banda_idBanda`)
    REFERENCES `mydb`.`Banda` (`idBanda`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_BandaMembro_Membro1`
    FOREIGN KEY (`Membro_idMembro`)
    REFERENCES `mydb`.`Membro` (`idMembro`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_BandaMembro_EstadoMembro1`
    FOREIGN KEY (`EstadoMembro_idEstadoMembro`)
    REFERENCES `mydb`.`EstadoMembro` (`idEstadoMembro`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

5.3. Análise de transações

O modelo, para preencher os requisitos, deve cumprir as seguintes *queries*:

a) Dado uma música, determinar a que álbum pertence;

```
SELECT DISTINCT A.titulo AS TítuloÁlbum,
                                B.nome AS Banda,
                                A.dataLancamento AS DataLançamento
FROM mydb.Album AS A
JOIN mydb.AlbumMusica AS AM
ON AM.Album_idAlbum = A.idAlbum
     JOIN mydb.Musica AS M
    ON M.idMusica = AM.Musica_idMusica
              JOIN mydb.BandaAlbum AS BA
         ON BA.Album_idAlbum = A.idAlbum
                       JOIN mydb.Banda AS B
              ON B.idBanda = BA.Banda_idBanda
WHERE M.titulo = 'Homem Do Leme'
```

TítuloÁlbum	Banda	DataLançamento
Cerco	Xutos e Pontapés	1985-11-17
Dia De Concerto	Rio Grande	1998-01-01

Figura 8: Resultados da transação a).

b) Dado uma música, determinar a que banda(s) pertence;

```
SELECT DISTINCT B.nome AS NomeBanda,
```

```
                                A.titulo AS Álbum
FROM mydb.Banda AS B
JOIN mydb.BandaAlbum AS BA
ON BA.Banda_idBanda = B.idBanda
      JOIN mydb.Album AS A
    ON A.idAlbum = BA.Album_idAlbum
               JOIN mydb.AlbumMusica AS AM
         ON AM.Album_idAlbum = A.idAlbum
                        JOIN mydb.Musica AS M
              ON M.idMusica = AM.Musica_idMusica
WHERE M.titulo = 'Homem Do Leme'
```

NomeBanda	Álbum
Xutos e Pontapés	Cerco
Rio Grande	Dia De Concerto

Figura 9: Resultados da transação b).

c) Dado um álbum, determinar que músicas este contém;

```
SELECT DISTINCT M.titulo AS Título, M.duracao AS Duração
FROM mydb.Album AS A
JOIN mydb.AlbumMusica AS AM
ON AM.Album_idAlbum = A.idAlbum
      JOIN mydb.Musica AS M
    ON M.idMusica = AM.Musica_idMusica
WHERE A.titulo = 'Dia De Concerto'
```

Título	Duração
A Fisga	01:43:00
Fui Às Sortes e Safei-me	03:06:00
O Sobrescrito	03:36:00
Postal Dos Correios	02:40:00
Menina Estás À Janela	03:12:00
A Paixão (Segundo Nicolau Da Viola)	04:11:00
Negras Como A Noite	05:19:00
Fado Triste	03:59:00
A Ilha	05:30:00
Saudade	03:01:00
Queda Do Impéri0	02:30:00
A História De Zé Passarinho	02:41:00
Quem És Tu, De Novo	03:17:00
Os Loucos De Lisboa	03:38:00
Deixa-me Rir	05:23:00
Homem Do Leme	04:01:00
Porto Sentido	04:34:00

Figura 10: Resultados da transação c).

d) Dado um álbum, determinar a que banda(s) pertence;

```
SELECT DISTINCT B.nome AS NomeBanda
FROM mydb.Banda AS B
JOIN mydb.BandaAlbum AS BA
ON BA.Banda_idBanda = B.idBanda
     JOIN mydb.Album AS A
    ON A.idAlbum = BA.Album_idAlbum
WHERE A.titulo = 'XIII'
```

NomeBanda
Xutos e Pontapés

Figura 11: Resultados da transação d).

e) Dado uma banda, determinar a lista de músicas da mesma;

```
SELECT DISTINCT M.titulo AS Título,
                               M.duracao AS Duração,
                A.titulo AS Álbum
FROM mydb.Musica AS M
JOIN mydb.AlbumMusica AS AM
ON AM.Musica_idMusica = M.idMusica
     JOIN mydb.Album AS A
    ON A.idAlbum = AM.Album_idAlbum
            JOIN mydb.BandaAlbum AS BA
        ON BA.Album_idAlbum = A.idAlbum
                    JOIN mydb.Banda AS B
            ON B.idBanda = BA.Banda_idBanda
WHERE B.nome = 'Pólo Norte';
```

Título	Duração	Álbum
A Fisga	01:43:00	Rio Grande
A Fisga	01:43:00	Dia De Concerto
O Caçador Da Adiça	03:14:00	Rio Grande
Fui Às Sortes e Safei-me	03:06:00	Rio Grande
Fui Às Sortes e Safei-me	03:06:00	Dia De Concerto
O Dia Do Nó	02:25:00	Rio Grande
O Sobrescrito	03:36:00	Rio Grande
O Sobrescrito	03:36:00	Dia De Concerto
Lisnave	03:55:00	Rio Grande
Dia De Passeio	04:09:00	Rio Grande
Postal Dos Correios	02:40:00	Rio Grande
Postal Dos Correios	02:40:00	Dia De Concerto
Senta-te Aí	02:48:00	Rio Grande

Figura 12: Resultados da transação e).

f) Dado uma banda, determinar a lista de álbuns da mesma;

```
SELECT DISTINCT A.titulo AS Título,
        A.idAlbum AS ID,
        B.nome AS NomeBanda,
        A.capa AS Capa,
        A.dataLancamento AS DataLançamento,
```

```
        A.editora AS Editora,
        A.descricao AS Descrição
FROM mydb.Banda AS B
JOIN mydb.BandaAlbum AS BA
ON BA.Banda_idBanda = B.idBanda
     JOIN mydb.Album AS A
    ON A.idAlbum = BA.Album_idAlbum
WHERE B.nome = 'Pólo Norte'
```

Título	ID	NomeBanda	Capa	DataLançamento	Editora	Descrição
Pólo Norte ao Vivo	1	Pólo Norte	NULL	2000-08-24	BMG	NULL
Jogo da Vida	2	Pólo Norte	NULL	2002-04-01	BMG	NULL
Deixa o Mundo Girar	3	Pólo Norte	NULL	2005-11-01	Lemon	NULL
15 Anos	4	Pólo Norte	NULL	2008-04-01	Steve Lyon	NULL

Figura 13: Resultados da transação f).

g) Dado uma banda, determinar a lista de géneros da mesma;

```
SELECT DISTINCT G.nome AS Género
FROM mydb.Banda AS B
JOIN mydb.BandaGenero AS BG
ON BG.Banda_idBanda = B.idBanda
     JOIN mydb.Genero AS G
    ON G.idGenero = BG.Genero_idGenero
WHERE B.nome = 'Xutos e Pontapés'
```

Género
Pop rock
New wave
Punk rock
Folk rock

Figura 14: Resultados da transação g).

h) Dado uma banda, determinar a lista de membros atuais;

```
SELECT DISTINCT ME.nome AS Nome,
                EM.estado AS Estado
FROM mydb.Membro AS ME
JOIN mydb.BandaMembro AS BM
ON BM.Membro_idMembro = ME.idMembro
     JOIN mydb.Banda AS B
    ON B.idBanda = BM.Banda_idBanda
            JOIN mydb.EstadoMembro AS EM
        ON EM.idEstadoMembro = BM.EstadoMembro_idEstadoMembro
WHERE B.nome = 'Xutos e Pontapés'
```

	Nome	Estado
▶	Zé Pedro	Ativo
	Kalú	Ativo
	Tim	Ativo
	João Cabeleira	Ativo
	Gui	Ativo
	Zé Leonel	Inativo
	Francis	Inativo

Figura 15: Resultados da transação h).

i) Dado um membro, determinar em que banda(s) toca;

```
SELECT DISTINCT B.nome AS Banda,
                EM.estado AS Estado
FROM mydb.Membro AS ME
JOIN mydb.BandaMembro AS BM
ON BM.Membro_idMembro = ME.idMembro
     JOIN mydb.Banda AS B
    ON B.idBanda = BM.Banda_idBanda
            JOIN mydb.EstadoMembro AS EM
        ON EM.idEstadoMembro = BM.EstadoMembro_idEstadoMembro
WHERE ME.nome = 'Tim'
```

	Banda	Estado
▶	Xutos e Pontapés	Ativo
	Rio Grande	Inativo

Figura 16: Resultados da transação i).

j) Dado um género, determinar a lista de bandas do mesmo.

```
SELECT DISTINCT B.nome AS Banda
FROM mydb.Banda AS B
JOIN mydb.BandaGenero AS BG
ON BG.Banda_idBanda = B.idBanda
     JOIN mydb.Genero AS G
    ON G.idGenero = BG.Genero_idGenero
            JOIN mydb.Genero AS G2
        ON G2.idGenero = G.Genero_idGenero
WHERE G.nome = 'Pop' OR G2.nome = 'Pop';
```

	Banda
▶	Pólo Norte
	Xutos e Pontapés
	Rio Grande

Figura 17: Resultados da transação j).

5.4. Estimativa de requisitos de espaço de disco

Para estimar o espaço da base de dados e mostrar a importância da normalização vamos mostrar vários exemplos.

Considerando que uma imagem (LONGBLOB) ocupa 20 000 bytes em média, e o tipo “Text” ocupa no máximo 64000 bytes, a entidade “Banda”, quando não-normalizada, ou seja, sendo o “Estado”, “Pais” e “Membros” seus atributos, ocupa 148 187 bytes por tuplo. Um tuplo da entidade “Banda” normalizada ocupa 148 115 bytes. Para podermos comparar completamente, um tuplo da entidade “Estado” ocupa 24 bytes, um tuplo da entidade “Pais” ocupa 34 bytes e um tuplo da entidade “Cidade” ocupa 38 bytes.

Vamos considerar, no cálculo da entidade estar normalizada, 20 cidades, 1 país e 4 estados.

Então, consideremos os seguintes exemplos:

- Para 100 bandas, a diferença entre não estar normalizada e estar normalizada, neste caso é de aproximadamente: 6 310 bytes.
  ((628187*100) - (628115*100+20*38+34+24*4));
- Para 10 000 bandas, a diferença entre não estar normalizada e estar normalizada, neste caso é de aproximadamente: 702 KBytes;
- Para 1 000 000 bandas, a diferença entre não estar normalizada e estar normalizada, neste caso é de aproximadamente: 69 MBytes;
- Para 100 000 000 bandas, a diferença entre não estar normalizada e estar normalizada, neste caso é de aproximadamente: 6.7 GBytes.

Como podemos ver, o crescimento da diferença do espaço ocupado aumenta exponencialmente à medida que aumentamos o número de bandas para uma base de dados normalizada ou não. Normalizado, 1 milhão de bandas ocupa 138 GB, sendo 13% devido aos logótipos das bandas.

Entidade	Espaço máximo ocupado em Bytes (por tuplo)	Número de tuplos
Banda	148115	3
BandaGenero	8	6
Genero	28	6
EstadoBanda	24	2
Cidade	38	20
Pais	34	1
BandaAlbum	8	12
BandaMembro	12	19
EstadoMembro	24	4
Membro	104	18
Album	84087	12
AlbumMusica	8	125
Musica	64037	125

Tabela 8: Espaço ocupado por tuplo de cada entidade.

Dado o exemplo representado pela tabela 8, o total ocupado é de 9.02 Mbytes.

5.5. Definição de vistas de utilizadores e regras de acesso

No nosso projeto, optámos por não usar diferentes tipos de utilizadores, pelo que a vista é a mesma para qualquer utilizador. Então, para qualquer utilizador que pretenda aceder à nossa base de dados, este pode aceder a toda a informação que guardámos, visto também que não existe informação que necessite de ser ocultada. O mesmo se passa para regras de acesso, pois visto que não existem diferentes tipos utilizadores, não é preciso aplicar regras de acesso.

No entanto, e numa implementação futura da BD para utilização num site, por exemplo, poderíamos ter a necessidade de criar utilizadores normais e administradores. Considerando esta hipótese, teríamos de criar entidade que regessem estes dois tipos de utilizadores, e aqui sim teríamos de aplicar regras de acesso. Por exemplo, administradores poderiam ver informações pessoais de outros utilizadores, acrescentar/remover bandas, membros, ou seja, geriam o cerne da informação da base de dados. Por outro lado, utilizadores normais poderiam aceder à informação musical disponibilizada pela base de dados e ao seu perfil, não podendo, no entanto, modificar a informação musical disponibilizada. Também poderíamos fazer com que

estes utilizadores pudessem aceder à informação pessoal de outros utilizadores, mas apenas poderiam aceder a algumas destas, como o nome, data de nascimento, etc.

6. Conclusão e trabalhos futuros

Na unidade curricular de Bases de Dados, decidimos o tema do nosso trabalho, e, a partir daí, comprometemo-nos a fazer uma base de dados que solucionasse o nosso tema.

O tema escolhido, era algo que já todos os elementos do nosso grupo tinham pensado em fazer extra-curricularmente, mas não tínhamos, de todo, o conhecimento necessário para sequer começar a desenvolver. Achamos então que esta era uma oportunidade ideal para pôr em prática o que vamos aprendendo ao longo deste semestre, sobre Bases de Dados, através de um trabalho que nos satisfaça, fugindo aos padrões que costumam ser os trabalhos propostos noutros contextos, que por vezes são enfadonhos.

Analisando o desenvolvimento do projeto, denotámos que para desenvolver uma base de dados é necessário proceder à modelação conceptual, lógica e física, para que no final tenhamos a certeza de que o trabalho está bem desenvolvido e que cumpre os requisitos definidos.

Bibliografia

Connolly, T., Begg, C., Database Systems, A Practical Approach to Design, Implementation, and Management, Addison-Wesley, 4ª Edição, 2004. ISBN-10: 0321210255. ISBN-13: 978-0321210258.

Gouveia, Feliz, Fundamentos de Bases de Dados, FCA-Editora de Informática, Lda., 2014. ISBN: 978-972-722-799-0

Lista de Siglas e Acrónimos

BD	Base de Dados
GB	Gigabytes
SGDB	Sistema de gestão de bases de dados