Projeto Final BD

1 Introdução

Quando há requisito de eficiência nas organizações, tanto com fins lucrativos ou não, é importante que haja conhecimento sobre os dados que transitam por tal organização. Tais dados devem ser trabalhados de modo a se adquirir informações úteis para um bom planejamento, do gerencial ao estratégico.

É possível armazenar tais dados em memória simples de um computador, sendo esta um HD, e realizar todas as operações sobre tais dados; mas há o caso de que, cada vez mais, a massa de dados vem aumentado. Tal aumento exige um gerenciamento adequado, promovido pelos *Sistemas Gerenciadores de Banco de Dados* (*SGBDs*).

Primeiramente, é preciso compreender o que seria um *banco de dados* (BD): pode ser visto como o equivalente eletrônico de um armário de arquivamento, uma vez que é a coleção de dados persistentes utilizadas pelos sistemas de aplicação de uma empresa. Dessa forma, um SGBD é responsável pelo gerenciamento dessas coleções, garantindo integridade, segurança e outras características.

Os dados podem ser extraídos de qualquer fonte, mas, para este trabalho, será utilizada a *INDA* (*Infraestrutura Nacional de Dados Abertos*); visto que são dados que estão à disposição livremente a todos, mas apenas alguns a acessam. Dessa forma, ao se trabalhar com tais dados e gerar informações consideráveis estaríamos exercendo a nossa cidadania, buscando discrepâncias entre os dados e apontando algumas irregularidades.

Como escopo, foi escolhido os dados sobre **Diárias e passagens** (Tais dados estão disponível *online*: `http://www.portaltransparencia.gov.br/`) limitados ao período: $jan/2015$ a $jun/2015$, visando responder os seguintes questionamentos:

1. Qual o gasto total em cada mês?
2. Quais os órgãos que mais gastaram?
3. Quais os programas que mais gastaram?
4. Quais os servidores que mais gastaram?

5. Quais as funções que mais gastaram?

2 Diagrama Entidade Relacionamento (DER)

Um DER constitui uma forma de representação gráfica para os conceitos atribuídos ao *Modelo Entidade Relacionamento* (MER), sendo este um modelo de dados conceitual de auto nível, visto que está centrado na percepção dos usuários sobre os dados, não importando a maneira na qual os dados serão armazenados. Dessa forma, entendisse DER como representação visual sobre os dados que serão tratados no BD.

Como os dados fornecidos vem em forma de planilhas, foi preciso realizar uma leitura e uma interpretação dos mesmos, tentando identificar o que veriam a ser as entidades e os relacionamentos para o nosso BD. Dessa forma, obteve-se a Figura 1 como resutaldo do estudo realizado.

3 Modelo Relacional (MR)

Um MR representa os dados num BD como uma coleção de relações, denominadas de tabelas. Essa coleção será implantada no SGBD, ou seja, o MR representa a construção física do BD. Por isso é desenvolvido a partir do DER e, no escopo deste trabalho, tem-se a Figura 2 como resultado.

4 Avaliação das formas normais

A normalização é importante para identificar um bom projeto relacional. Um bom MER e sua consequente conversão para um MR, praticamente, deixa o esquema relacional *normalizado* .

Ao se tratar de normalização, consideram-se três formais normais, onde: na primeira (1FN) há a caracterização de um valor de uma coluna de uma tabela é indivísel; na segunda (2FN), se estiver na 1FN e todo atributo do complemento de uma chave candidata é totalmente funcionalmente dependente daquela chave; for fim, na terceira, se estiver na 2FN e todos os atributos não-chave forem dependentes não-transitivos da chave primária.

Dessa forma, foram selecionadas três tabelas: Pessoa, Ação e Função; onde:

- **Pessoa:**

Cpf	Nome	Cod_superior	Cod_subordinado	Cod_unidade_gestora

  - 1FN: Todos os valores das colunas da tabela são indivisíveis;
  - 2FN: Chave candidata: Cpf

$$\begin{aligned} Cpf &\rightarrow Nome \\ Cpf &\rightarrow Cod\_superior \\ Cpf &\rightarrow Cod\_subordinado \\ Cpf &\rightarrow Cod\_unidade\_gestora \end{aligned}$$

  - 3FN: Não há nenhuma transitividade nas colunas, uma vez que todos são definidos unicamente somente pelo Cpf.

- **Ação:**

Codigo	Nome	Linguagem_cidada

  - 1FN: Todos os valores das colunas da tabela são indivisíveis;
  - 2FN: Chave candidata: Código

$$\begin{aligned} Codigo &\rightarrow Nome \\ Codigo &\rightarrow Linguagem\_cidada \end{aligned}$$

  - 3FN: Não há nenhuma transitividade nas colunas, uma vez que todos são definidos unicamente somente pelo Código.

- **Função:**

Codigo	Nome

- 1FN: Todos os valores das colunas da tabela são indivisíveis;
- 2FN: Chave candidata: Código

$$Codigo \rightarrow Nome$$

- 3FN: Não há nenhuma transitividade nas colunas, uma vez que todos são definidos unicamente somente pelo Código.

5 Criação do BD

A partir do MR construído, foi realizado o processo da engenhaia reversa, de modo a ser obter o *script* corresponde à criação daquele BD. Como resultado tem-se, para apenas uma tabela, o *script* demonstrado no Listing 1.

```
1  DROP DATABASE IF EXISTS 'trab_bd'
2  CREATE DATABASE  IF NOT EXISTS `trab_bd` /*!40100 DEFAULT CHARACTER SET latin1 */;
3  USE `trab_bd`;
4  -- MySQL dump 10.13  Distrib 5.5.53, for debian-linux-gnu (x86_64)
5  --
6  -- Host: 127.0.0.1    Database: trab_bd
7  -- ------------------------------------------------------
8  -- Server version 5.5.53-0ubuntu0.14.04.1
9
10 /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
11 /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
12 /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
13 /*!40101 SET NAMES utf8 */;
14 /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
15 /*!40103 SET TIME_ZONE='+00:00' */;
16 /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
17 /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
18 /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
19 /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
20
21 --
22 -- Table structure for table `ORGAO_SUPERIOR`
23 --
24
25 DROP TABLE IF EXISTS `ORGAO_SUPERIOR`;
26 /*!40101 SET @saved_cs_client     = @@character_set_client */;
```

```
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `ORGAO_SUPERIOR` (
  `codigo` int(255) NOT NULL DEFAULT '0',
  `nome` varchar(255) NOT NULL,
  PRIMARY KEY (`codigo`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `ORGAO_SUPERIOR`
--

LOCK TABLES `ORGAO_SUPERIOR` WRITE;
/*!40000 ALTER TABLE `ORGAO_SUPERIOR` DISABLE KEYS */;
/*!40000 ALTER TABLE `ORGAO_SUPERIOR` ENABLE KEYS */;
UNLOCK TABLES;
```

Listing 1: Trecho referente à criação do BD e de uma tabela no MySQL.

6 Processo de ETL

Como já foi anteriormente na Seção 2, os dados obtidos a partir da INDA são planilhas e para elas é preciso realizar um processo, de forma a se obter dados interpretáveis pelo SGBD utilizado, que no caso foi o *MySQL Workbench*. Tal processo é denominado *Processo ETL* (*Extract*, *Transform*, *Load*), pois nele há a extração dos dados existentes nas planilhas, há uma manipulação para formatar a inserção no SGBD e, por fim, há o povoamento do BD com os dados extraídos das planilhas.

As primeiras duas etapas do processo foram realizadas por um programa me linguagem JAVA, enquanto que a última era realizada diretamente no SGBD. Como resultado da primeira parte há um *script* .sql com todas as inserções, tal *script* é executado na segunda parte.

Todas as planilhas tem os mesmos tipos de dados e nas mesmas colunas, num total de 21 colunas. Com este conhecimento e sabendo o MR, que deveria ser implementado, a extração ocorreu de forma a respeitar tais conhecimento e os relacionados à Seção 4, tendo os atributos-chave únicos para cada registro. Na etapa de transformação, há a formação do comando de inserção, que segue a sintaxe do Listing 2.

```
INSERT INTO <tabela> (<lista de atributos>) VALUES (<lista de valores correspondentes>)
```

Listing 2: Trecho referente à sintaxe de inserção no MySQL.

7 Persistência e Visualização

A camada de persistência e o ambiente para visualização dos dados foram feitos com a plataforma *OutSystems* e está disponível no endereço: `https://diegoleite.outsystemscloud.com/Trab_BD/`

8 View, Procedure e Trigger

As *view* são tabelas virtuais criadas a partir de um *SELECT*. Normalmente, são utilizadas para prover segurança e facilidade de desenvolvimento. As *procedure* são programas que podem ser armazenados no SGBD, onde cada SGBD tem suas próprias linguagens. E as *triggers* é um objeto de banco de dados, associado a uma tabela, definido para ser disparado, respondendo a um evento em participar.

Dessa forma foram desenvolvidas os seguintes arquivos.

```
create view view1 as

select ACAO.nome, ACAO.codigo, PROGRAMA.nome, PROGRAMA.codigo,
ORGAO_SUBORDINADO.nome, ORGAO_SUBORDINADO.codigo,
UNIDADE_GESTORA.nome, UNIDADE_GESTORA.codigo, PESSOA.nome, PESSOA_PAG.valor
from ACAO
    inner join PROG_ACAO on
      ACAO.codigo = PROG_ACAO.cod_acao
    inner join PROGRAMA ON
      PROGRAMA.codigo = PROG_ACAO.cod_prog
    inner join PROG_ORGAO on
      PROGRAMA.codigo = PROG_ORGAO.cod_prog
    inner join ORGAO_SUBORDINADO ON
      ORGAO_SUBORDINADO.codigo = PROG_ORGAO.cod_sub
    inner join UNIDADE_GESTORA ON
      UNIDADE_GESTORA.cod_subordinado = ORGAO_SUBORDINADO.codigo
    inner join PESSOA ON
      PESSOA.codigo_unidade_gestora = UNIDADE_GESTORA.codigo
    inner join PESSOA_PAG ON
      PESSOA.cpf = PESSOA_PAG.cpf_pessoa
    inner join PAGAMENTO ON
      PAGAMENTO.documento_pagamento = PESSOA_PAG.doc_pag
```

Listing 3: Trecho referente à *view* desenvolvida.

```
delimiter $$
create procedure sp_lista_clientes(in opcao integer)
```

```
begin
    if opcao == 0 then
        select PESSOA.nome, sum(PESSOA_PAG.valor) as total from PESSOA inner join
PESSOA_PAG on PESSOA.cpf = PESSOA_PAG.cpf_pessoa inner join PAGAMENTO
on PAGAMENTO.documento_pagamento = PESSOA_PAG.doc_pag where total > 5000 group by PESSOA.nome
    else
        if opcao == 1 then
                    select PESSOA.nome, sum(PESSOA_PAG.valor) as total from PESSOA inner join
PESSOA_PAG on PESSOA.cpf = PESSOA_PAG.cpf_pessoa inner join PAGAMENTO
on PAGAMENTO.documento_pagamento = PESSOA_PAG.doc_pag where total > 10000 group by PESSOA.nome

        end if;
    end if;
  end $$
delimiter ;
```

Listing 4: Trecho referente ao *procedure* desenvolvido.

```
CREATE
[DEFINER = { user | CURRENT_USER}]
TRIGGER trigger1 AFTER
insert on PESSOA_PAG FOR EACH ROW update PESSOA_PAG SET valor = round(PESSOA_PAG.valor)
where cpf_pessoa in (select cpf from PESSOA)
```

Listing 5: Trecho referente ao *trigger* desenvolvido.

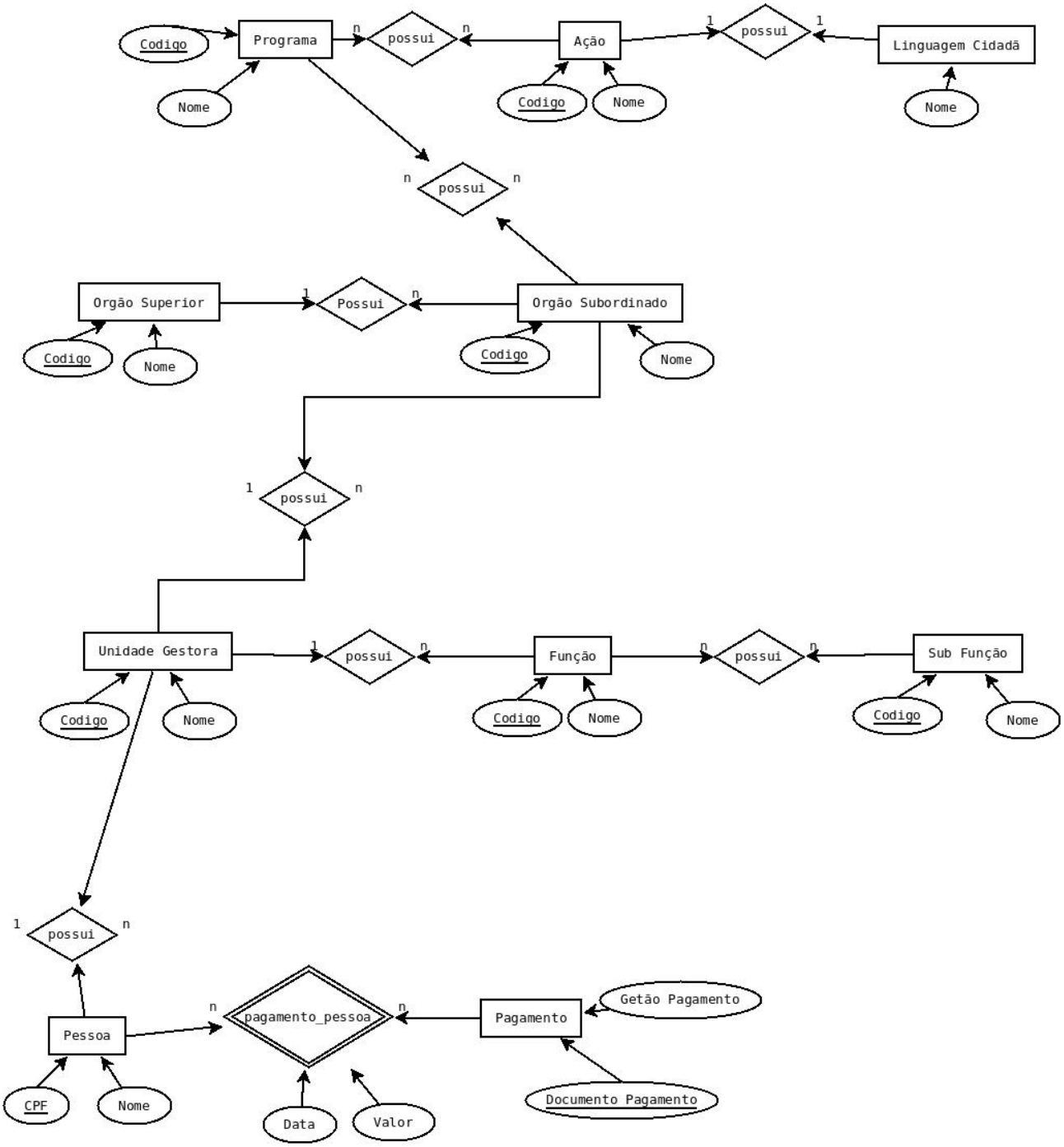

Figura 1: DER gerado a partir das tabelas e nossa interpretação.

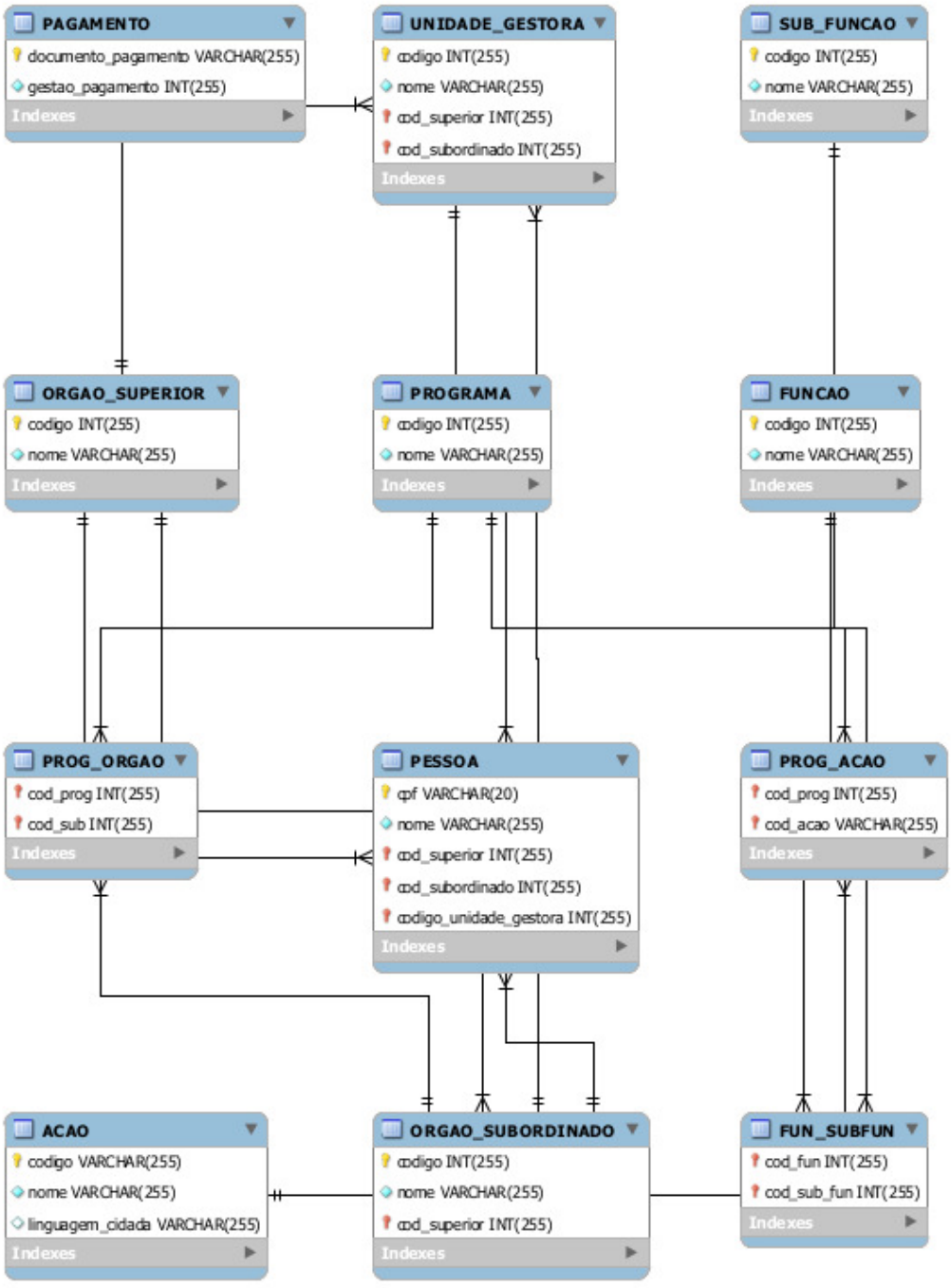

Figura 2: MR gerado a partir do DER.