Leonardo Fonseca Pinheiro 11219241

a) As inconsistências e perda de dados estão sendo causadas pelo mal desenvolvimento da estrutura da base de dados. Além de não respeitar a primeira diretriz, colocando atributos em uma tabela na qual eles não devem estar, o modelo atual tem redundância nas tuplas, permitindo a ocorrência de inúmeras anomalias. Isso ocorre porque a chave envolvendo CPF e Projeto faz com que o mesmo funcionário seja cadastrado uma vez para cada projeto, gerando a repetição de todos os dados pessoais e deixando a base vulnerável para dados incoerentes, como por exemplo, um mesmo CPF com dois nomes diferentes vinculados a ele. Ademais, a relação entre categoria e código de departamento em Projeto pode gerar uma tupla ilegítima.

Por fim, os elementos não são definidos como NotNull, permitindo que o nome de seu funcionário, por exemplo, seja nulo. Ou seja, o modelo atual não segue as diretrizes, deixando a base muito exposta a inconsistências e perda de dados.

b) Para essa solução, a FK Projeto, contida em empregado, será extinguida e tratada no fim do exercício, uma vez que a solução atual traz problemas, mas ainda é importante garantir que um empregado possa participar de mais de um projeto.

Segundo o texto e o esquema, as relações funcionais são:

CPF -> Nome, DataNasc, Endereço, Atividade, Categoria, CodigoDepto, ChefeDepto, NomeDepto

Atividade -> Categoria, CodigoDepto, ChefeDepto, NomeDepto

Categoria -> CodigoDepto, ChefeDepto, NomeDepto

CodigoDepto -> ChefeDepto, NomeDepto (Assume que o código é único, mas o chefe e o nome não)

CodigoProj -> NomeProj, CodigoDepto, Categoria

NomeProj -> CodigoDepto, Categoria

Empregado ->> Projetos (É considerado que o chefe é um empregado)

Com base nisso, a primeira coisa a ser é garantir a 1ª Forma Normal, deixando todos os atributos atômicos e monovalorados, com as seguintes mudanças:

CPF -> Nome, DataNasc, Rua, Nro, Cidade, Estado, CEP, Atividade, Categoria, CodigoDepto, ChefeDepto, NomeDepto

Com o passo anterior, garantimos também a 2ª Forma Normal, uma vez que ela está na 1ª Forma Normal e todos os atributos não primários possuem dependência total, transitiva ou não, da chave primária.

Então, precisamos tratar os casos de transitividade, para garantir a 3ª Forma Normal.

Etapas:

1. Empregado = {Nome, CPF, DataNasc, Rua, Nro, Cidade, Estado, CEP, Atividade, Categoria, CodigoDepto, ChefeDepto, NomeDepto}

---

2. Empregado = {Nome, CPF, DataNasc, Rua, Nro, Cidade, Estado, CEP, Atividade}

   Ativ = {Atividade, Categoria, CodigoDepto, ChefeDepto, NomeDepto}

---

3. Empregado = {Nome, CPF, DataNasc, Rua, Nro, Cidade, Estado, CEP, Atividade}

   Ativ = {Atividade, Categoria}

   Cat = {Categoria, CodigoDepto, ChefeDepto, NomeDepto}

---

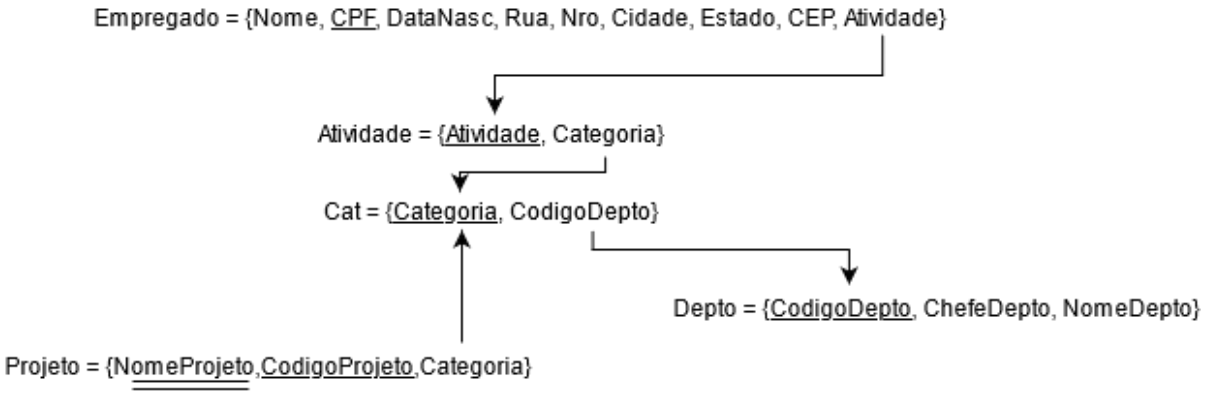
4. Empregado = {Nome, CPF, DataNasc, Rua, Nro, Cidade, Estado, CEP, Atividade}

   Ativ = {Atividade, Categoria}

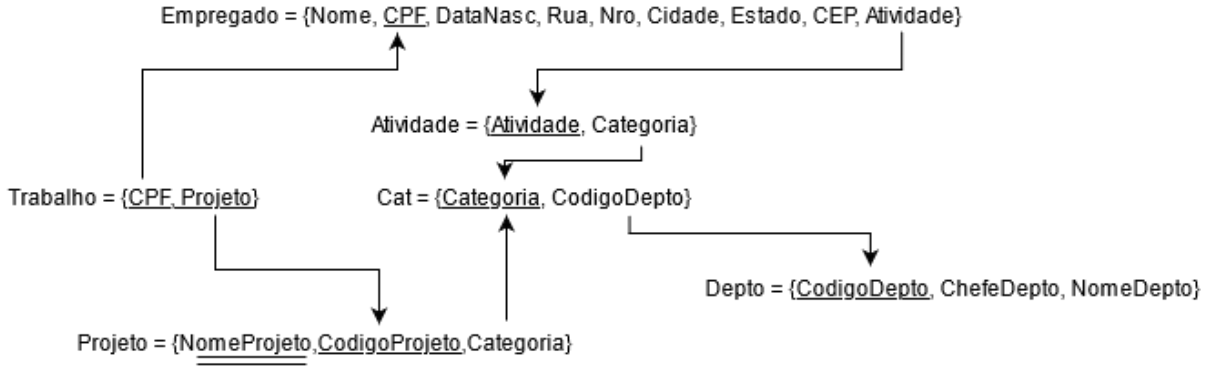
   Cat = {Categoria, CodigoDepto}

   Depto = {CodigoDepto, ChefeDepto, NomeDepto}

Como podemos ver, ao normalizar Empregado, simplificamos nosso processo na simplificação do Projeto, porque gerou a relação Cat, que tem Categoria como chave. Essa etapa teria que ser feita para o Projeto, mas como já foi feita, temos as relações normalizadas na imagem abaixo.



Por fim, como dito no início do item b, é necessário tratar a relação entre Empregado e Projeto, que foi extinguida. Como foi estabelecido na lista de relações, ela será tratada como uma dependência Multivalorada, como na imagem abaixo, que possui o modelo final.



Dessa forma, temos a solução completamente normalizada, garantindo consistência muito superior à anterior. Contudo, devido ao aumento de 2 para 6 tabelas, teremos uma redução na eficiência da busca, porque será necessário passar por mais operações de join antes de obter todas as informações e isso requer tempo. Mas dada a situação da consistência anteriormente e a necessidade de dados corretos nessa situação, esse trade-off é válido.