Etapas de Normalização de Dados

A normalização de banco de dados é um processo de organização de dados em um banco de dados relacional, a fim de reduzir a redundância de dados e evitar inconsistências. Isso é alcançado por meio da aplicação de regras de normalização que descrevem como dividir as informações em tabelas separadas, de forma que cada tabela contenha dados exclusivos e que possam ser relacionados a outras tabelas por meio de chaves estrangeiras.

Supondo inicialmente que modelamos uma entidade para matrícula de um aluno e obtivemos o seguinte resultado:

tabela_matrícula

cod_aluno	cod_turma	cod_disciplina	nome_diciplina	nome_aluno	cod_local_nasc	nome_local_nasc
111	001	1	Logica de programação	Alex	94	Uberlândia/MG
111	001	2	Banco de dados	Alex	94	Uberlândia/MG
222	002	3	Java Script	Ari	88	Indaiatuba/SP
222	002	4	POO	Ari	88	Indaiatuba/SP
333	001	2	Banco de dados	Ana	90	São Paulo/SP
333	001	4	POO	Ana	90	São Paulo/SP
444	002	1	Logica de programação	Pedro	75	Brasilian/DF

Uma vez obtido o resultado da tabela de matrícula iremos aplicar todas as regras de normalização na mesma.

Verificação 1FN(Primeira forma normal)

Na primeira forma normal temos as seguinte regras:

- A relação não pode possuir atributos compostos ou multivalorados

Ao analisarmos a entidade é observado que o atributo **nome_local_nascimento** é um atributo composto e pode ser dividido em 2 atributos cidade e UF

tabela_matrícula

cod_aluno	cod_turma	cod_disciplina	nome_diciplina	nome_aluno	cod_local_nasc	nome_local_nasc
111	001	1	Logica de programação	Alex	94	Uberlândia/MG
111	001	2	Banco de dados	Alex	94	Uberlândia/MG
222	002	3	Java Script	Ari	88	Indaiatuba/SP
222	002	4	POO	Ari	88	Indaiatuba/SP
333	001	2	Banco de dados	Ana	90	São Paulo/SP
333	001	4	POO	Ana	90	São Paulo/SP
444	002	1	Logica de programação	Pedro	75	Brasília/DF

Aplicando os passos 1FN temos o Resultado

tabela_matrícula

cod_aluno	cod_turma	cod_disciplina	nome_diciplina	nome_aluno	cod_local_nasc	cidade	UF
111	001	1	Logica de programação	Alex	94	Uberlândia	MG
111	001	2	Banco de dados	Alex	94	Uberlândia	MG
222	002	3	Java Script	Ari	88	Indaiatuba	SP
222	002	4	POO	Ari	88	Indaiatuba	SP
333	001	2	Banco de dados	Ana	90	São Paulo	SP
333	001	4	POO	Ana	90	São Paulo	SP
444	002	1	Logica de programação	Pedro	75	Brasília	DF

Verificação 2FN (Segunda forma normal)

- Para relações onde a chave primária é composta, não pode haver dependência funcional parcial em relação a chave.

Analise :

- O **codigo da diciplina** é dependente funcional de **codigo do aluno** e **codigo da turma**: (cod_aluno, cod_turma) -> cod_diciplina
- O nome do aluno tem uma dependência parcial pois só depende de **codigo do aluno**: (cod_aluno) -> nome_aluno

Seguindo o padrão da notação acima temos:
cod_aluno -> cod_local_nasc, cidade, UF

Obtida as análises precisamos separar esses atributos em outra entidade.

tabela_matrícula

cod_aluno	cod_turma	cod_disciplina	nome_diciplina	nome_aluno	cod_local_nasc	cidade	UF
111	001	1	Logica de programação	Alex	94	Uberlândia	MG
111	001	2	Banco de dados	Alex	94	Uberlândia	MG
222	002	3	Java Script	Ari	88	Indaiatuba	SP
222	002	4	POO	Ari	88	Indaiatuba	SP
333	001	2	Banco de dados	Ana	90	São Paulo	SP
333	001	4	POO	Ana	90	São Paulo	SP
444	002	1	Logica de programação	Pedro	75	Brasília	DF

Aplicando os passos 2FN temos o Resultado

tabela_aluno

cod_aluno	nome_aluno	cod_local_nas	cidade	UF
111	Alex	94	Uberlândia	MG
222	Ari	98	Indaiatuba	SP
333	Ana	90	São Paulo	SP
444	Pedro	75	Brasilia	DF

tabela_matrícula

cod_aluno	cod_turma	cod_diciplina	nome_diciplina
111	001	1	Lógica de Programação
111	001	2	Banco de dados
222	002	3	Javascript
222	002	4	POO
333	001	2	Banco de dados
333	001	4	POO
444	002	2	Lógica de programação

Observação:
Comparando as relações resultante com a relação na 1 forma normal, podemos observar que vários problemas foram corrigidos como: Corrigir a anomalia no caso de um aluno não ter sido cadastrado em uma turma e também problemas de repetições de tuplas.

Verificação 3FN

- **A relação não pode ter atributos não chave que dependem do funcionamento de outros atributos não chave.**

tabela_aluno

cod_aluno	nome_aluno	cod_local_nas	cidade	UF
111	Alex	94	Uberlândia	MG
222	Ari	98	Indaiatuba	SP
333	Ana	90	São Paulo	SP
444	Pedro	75	Brasilia	DF

Analisando a relação acima percebemos que as tuplas cidade e UF dependem funcionalmente de cod_local_nas
cod_local_nas -> cidade, UF

logo precisamos separar também essas relações:

tabela_aluno

cod_aluno	nome_aluno	cod_local_nas
111	Alex	94
222	Ari	98
333	Ana	90
444	Pedro	75

tabela_local

cod_local_nas	cidade	UF
94	Uberlândia	MG
98	Indaiatuba	SP
90	São Paulo	SP
75	Brasilia	DF

Novamente evitamos com essa normalização a repetição das tuplas nesse caso a de cidade

Seguindo agora os passos para a relação matrícula e aplicando a 3FN

tabela_matrícula

cod_aluno	cod_turma	cod_diciplina	nome_diciplina
111	001	1	Lógica de Programação
111	001	2	Banco de dados
222	002	3	Javascript
222	002	4	POO
333	001	2	Banco de dados
333	001	4	POO
444	002	2	Lógica de programação

Seguindo a análise percebemos que cod_disciplina -> nome_disciplina, logo, teremos que também separa esses atributos em outra entidade

Relações resultantes:

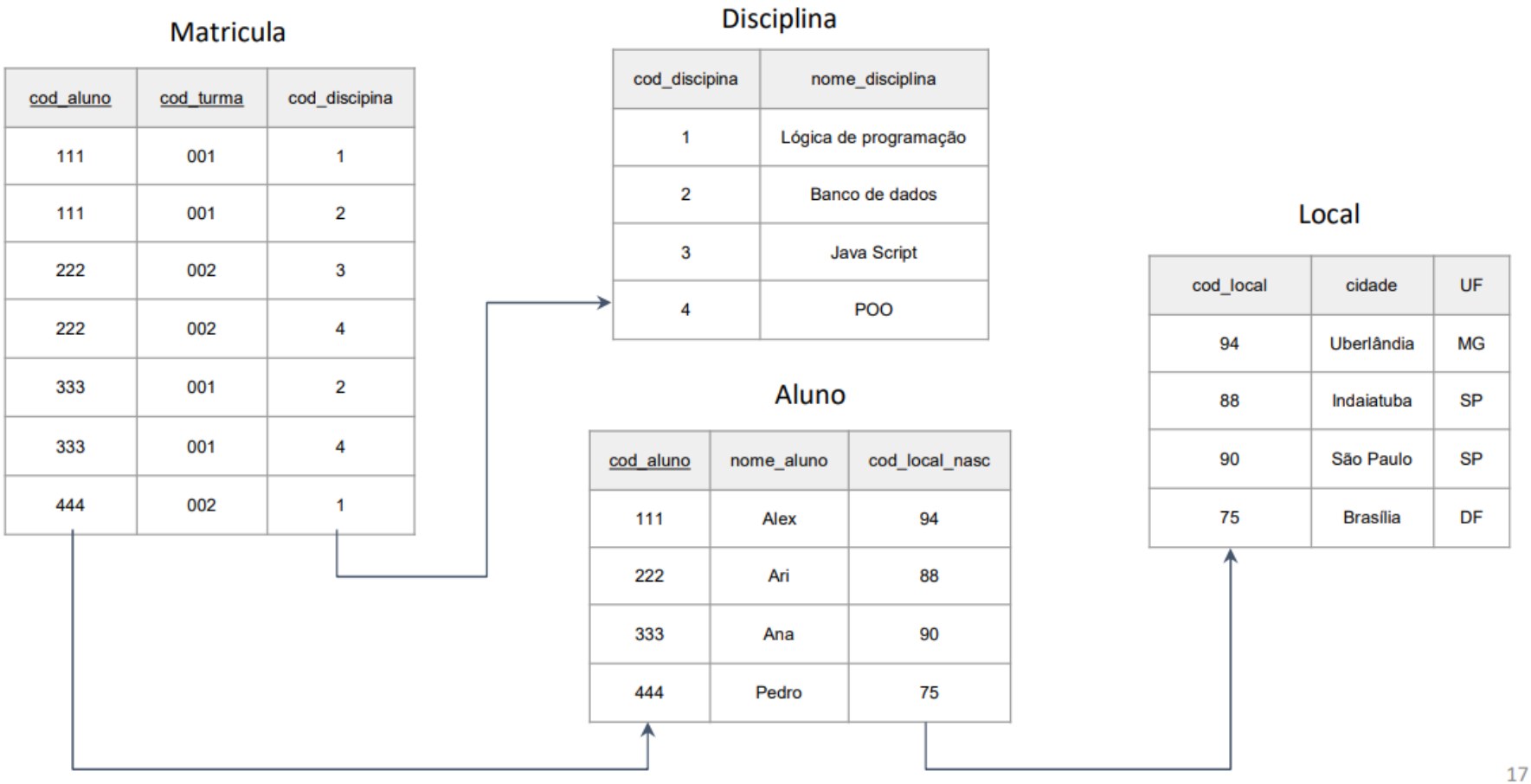
tabela_matrícula

cod_aluno	cod_turma	cod_diciplina
111	001	1
111	001	2
222	002	3
222	002	4
333	001	2
333	001	4
444	002	2

tabela_disciplina

cod_diciplina	nome_diciplina
1	Logica de Programação
2	Banco de dados
3	Java Script
4	POO

Resultado após a normalização:



Matricula

cod_aluno	cod_turma	cod_discipina
111	001	1
111	001	2
222	002	3
222	002	4
333	001	2
333	001	4
444	002	1

Disciplina

cod_discipina	nome_disciplina
1	Lógica de programação
2	Banco de dados
3	Java Script
4	POO

Aluno

cod_aluno	nome_aluno	cod_local_nasc
111	Alex	94
222	Ari	88
333	Ana	90
444	Pedro	75

Local

cod_local	cidade	UF
94	Uberlândia	MG
88	Indaiatuba	SP
90	São Paulo	SP
75	Brasília	DF