Qualidade de Esquemas Relacionais: Normalização

1. Motivação

A normalização é necessária (embora não suficiente) a um bom projeto relacional. Felizmente, um bom projeto de um esquema de entidades, e sua conseqüente conversão para um esquema relacional, segundo as regras vistas, praticamente deixa o esquema relacional *normalizado*.

Assim, utiliza-se a normalização somente para *validar* um projeto relacional.

Para entender o que a normalização significa, vamos dar primeiramente um exemplo de motivação.

HABILIDADES-ESPORTIVAS

Identidade	Nome	Endereço	Habilidade
8795835	Édson Arantes	Ponta da Praia	Futebol
8795835	Édson Arantes	Ponta da Praia	Voleibol
8795835	Édson Arantes	Ponta da Praia	Basquete
8795835	Édson Arantes	Ponta da Praia	Atletismo
8795835	Édson Arantes	Ponta da Praia	Tênis

Esta tabela é mal projetada! Diagnóstico: redundâncias desnecessárias.

Se Pelé mudar de endereço ? (*anomalia de atualização*)

Um novo esporte para Pelé ? (*anomalia de inclusão*)

Retirar Pelé do Banco de Dados (*anomalia de remoção*)

Idealmente:

HABILIDADES-ESPORTIVAS

Identidade	Nome	Endereço	Habilidade
8795835	Édson Arantes	Ponta da Praia	{Futebol, Voleibol, Basquete, Atletismo, Tênis}

Mas isto não é uma tabela (atributo *habilidade* não é atômico)! O que é possível fazer, dentro do modelo relacional?

ESPORTISTAS

Identidade	Nome	Endereço
8795835	Édson Arantes	Ponta da Praia
...	...	...

HABILIDADES

Identidade	Esporte
8795835	Futebol
8795835	Voleibol
8795835	Basquetebol
8795835	Atletismo
8795835	Tênis

A repetição da coluna Identidade é uma redundância necessária

2. Primeira Forma Normal (1FN)

Toda tabela deve ser "minimamente" normalizada (1FN).
É o **primeiro passo** do processo de normalização. Elimina os atributos multivalorados e compostos, permitindo apenas atributos atômicos.

Tabela em 1FN: O valor de uma coluna de uma tabela é indivisível.

Ex.: Empregado

Matrícula	Nome	Cod Cargo	NomeCargo	CodProj	DataFim	Horas
120	João	1	Programador	01	17/07/95	37
120	João	1	Programador	08	12/01/96	12
121	Hélio	1	Programador	01	17/07/95	45
121	Hélio	1	Programador	08	12/01/96	21
121	Hélio	1	Programador	12	21/03/96	107
270	Gabriel	2	Analista	08	12/01/96	10
270	Gabriel	2	Analista	12	21/03/96	38
273	Silva	3	Projetista	01	17/07/95	22
274	Abraão	2	Analista	12	21/03/96	31
279	Carla	1	Programador	01	17/07/96	27
279	Carla	1	Programador	08	12/01/96	20
279	Carla	1	Programador	12	21/03/96	51
301	Ana	1	Programador	12	21/03/96	16
306	Manoel	3	Projetista	17	21/03/96	67

A chave primária para a tabela empregados é (Matrícula,CodProj)

☹ Vimos que um dos objetivos da normalização é reduzir a redundância de dados, porém com a tabela acima aumentamos a redundância ?!?!

☺ Precisamos realizar outros passos de normalização para termos um bom projeto.

A 1FN possui características indesejáveis!

☹ **Anomalias da 1FN**
Inserção: não podemos inserir um empregado sem que este esteja alocado num projeto, nem um projeto sem que haja um empregado trabalhando nele (integridade de entidade).

Remoção: se precisarmos remover um projeto, as informações de empregados que estiverem lotados apenas naquele projeto serão perdidas.

Atualização: se um empregado for promovido de cargo teremos que atualizar os atributos CodCargo e NomeCargo em todas as tuplas nas quais aquele empregado está presente.

Conclusão:

Uma tabela em 1FN não evita, porém, anomalias de inclusão, atualização, e remoção. É preciso uma normalização mais "fina" , ou outras formas formas normais.

Segunda Forma Normal (2FN)
Terceira Forma Normal (3FN)

Esta normalização "fina" utiliza o conceito de *dependência funcional*

3. Dependências Funcionais

A → B, lê - se:

A *funcionalmente determina* B

B é *funcionalmente dependente* de A

B é *função* de A

Para cada valor de A só existe um valor de B.

A ¬→ B, negação de A → B.

A ou B podem ser um conjunto de atributos.

Identidade → Nome

Identidade → Endereço

Identidade ¬→ Habilidade

Nome ¬→ Identidade

Endereço ¬→ Identidade

Habilidade ¬→ Identidade

Identidade → Nome, Endereço

Idéia de normalização "fina": agrupar numa tabela somente dois conjuntos de atributos X e Y,
com X → Y.
X é então a *chave* da tabela, e Y é *complemento da chave*.

Conseqüência das definições de dependência funcional e de chave:
se X é chave então cada valor de X é *único*, e, conseqüentemente, um valor de X identifica uma linha da tabela.

É importante salientar que mais de um atributo (ou conjunto de atributos) pode ser chave, isto é, pode-se ter vários X → Y, cada X sendo uma *chave candidata*.

4. Segunda Forma Normal (2FN)

Uma tabela está na Segunda Forma Normal (2FN) se ela é 1FN e todo atributo do complemento de uma chave candidata é totalmente funcionalmente dependente daquela chave.

A, B, C => D (D é *totalmente funcionalmente dependente* de {A, B, C}) se para todo valor de {A, B, C} só existe um valor de D, e se D não é funcionalmente dependente de A, ou B, ou C.

A	B	C	D
a1	b3	c1	d4
a1	b1	c1	d2
a1	b3	c1	d1

a1 b3 c1 d1 não pode existir

A ¬→ D
B ¬→ D
C ¬→ D

Uma relação está em 2FN se cada atributo que não compõe a chave primária é funcionalmente dependente de toda a chave primária (definição mais conhecida).

Uma relação está em 2FN se nenhum atributo que não compõe uma chave candidata. Trata-se portanto, de eliminar dependências parciais.

Exemplo 1:

ESPORTISTA (Identidade, Nome, Endereço, Esporte)

Chaves Candidatas	Complementos da Chave
Identidade	Nome, Endereço, Esporte
{Nome, Endereço}	Identidade, Esporte

Identidade → Nome

Identidade → Endereço

Identidade ¬→ Esporte

{Nome, Endereço} $\neg$=> Esporte

Conclusão: O atributo Esporte deve ser retirado da relação ESPORTISTA.

ESPORTISTA (Identidade, Nome, Endereço)

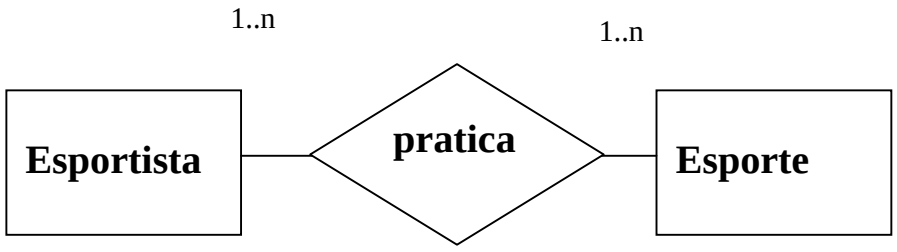
PRATICA-ESPORTE (Identidade, Esporte)

Um atributo sublinhado faz parte da chave.

Atualizar o endereço de Pelé: sem anomalia.

Incluir uma nova habilidade de Pelé: sem anomalia.

Note que estas tabelas podiam ser derivadas diretamente do esquema conceitual ER:



Ex2:
A tabela Empregado anterior após passarmos para 2FN resultaria em três tabelas:

Empregado

Matrícula	Nome	CodCargo	NomeCargo
120	João	1	Programador
121	Hélio	1	Programador
270	Gabriel	2	Analista
273	Silva	3	Projetista
274	Abraão	2	Analista
279	Carla	1	Programador
301	Ana	1	Programador
306	Manuel	3	Projetista

Projeto

CodProj	DataFim
01	17/07/95
08	12/01/96
12	21/03/96

Alocação

Matrícula	CodProj	Horas
120	01	37
120	08	12
121	01	45
121	08	21
121	12	107
270	08	10
270	12	78
273	01	22
274	12	31
279	01	27
279	08	20
279	12	51
301	01	16
301	12	85
306	12	67

☹ Anomalias da 2FN:

Inserção: Só podemos criar cargos se houver empregados designados para ele.

Remoção: Se removermos um empregado que ocupa unicamente um cargo na empresa, perderemos a informação deste cargo.

Atualização: Se um cargo muda de nome precisaremos mudar todas as tabelas em que este cargo aparece.

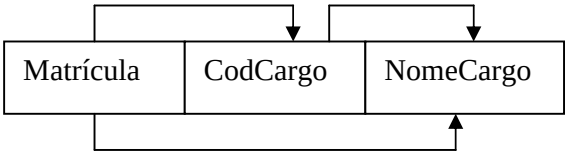
5. Terceira Forma Normal (3FN)

Envolve o conceito de dependência transitiva. Suponha que tenhamos uma tabela com colunas A, B e C.

Se a coluna C é funcionalmente dependente de B e B é funcionalmente dependente de A, então C é funcionalmente dependente de A.

Definição: Uma relação está em 3FN se, e somente se, estiver em 2FN e todos os atributos não-chave forem dependentes não-transitivos da chave primária.

Ex.: Ao analisarmos a nova tabela empregado que está em 2FN temos:



⇨ NomeCargo é dependente transitivo de Matrícula.

Removendo esta dependência transitiva, obteremos,além das tabelas Projeto e Alocação, as seguintes tabelas:

Empregado

Matrícula	Nome	CodCargo
120	João	1
121	Hélio	1
270	Gabriel	2
273	Silva	3
274	Abraão	2
279	Carla	1
301	Ana	1
306	Manuel	3

Cargo

CodCargo	Nome
1	Programador
2	Analista
3	Projetista

"Uma relação está em 3FN se todas as colunas da tabela são funcionalmente dependentes da chave inteira e nada além da chave".

A 3FN elimina as características mais potencialmente indesejáveis dos dados que estão em 2FN ou 1FN.

Existem outros casos especiais que requerem mais níveis de normalização: Boyce-Codd, 4FN e 5FN

6. Uma Metodologia de Normalização

Passo 1: Tome projeções de tabelas 1FN para eliminar todas as dependências funcionais não-totais. O resultado é uma coleção de tabelas 2FN.

Passo 2: Tome projeções das tabelas obtidas no passo 1 para eliminar todas as dependências transitivas. O resultado é uma coleção de relações 3FN.

1 FN: eliminação de atributos multivalorados e compostos.

2 FN: estar em 1FN e todos os atributos que não são chaves devem ser totalmente dependentes desta. Ou seja, não pode haver dependência parcial.

3 FN: estar em 1 FN e 2FN e os atributos não-chaves não podem depender de outros também não-chaves (dependência transitiva).