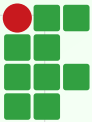

Instituto Federal Minas Gerais - Campus Bambuí
Departamento de Engenharia e Computação - DEC
Curso de Engenharia de Computação - ENGCOMP

Banco de Dados I

06 – Normalização

Marcos Roberto Ribeiro

2024

Introdução

- Sempre que trabalhamos com banco de dados relacionais, não desejamos redundâncias que podem levar a armazenamento desnecessário e inconsistências
- O armazenamento desnecessário de informações pode levar a problemas de desempenho devido ao grande número de dados armazenados em duplicidade
- As inconsistências causam a certas *anomalias* que podem prejudicar as informações do banco de dados

Anomalias em bancos de dados não normalizados

Anomalia de alteração: Se um dado em duplicidade é alterado, o banco de dados apresentará inconsistências a menos que todas as cópias do dados sejam atualizadas

Anomalia de inclusão: Não é possível incluir uma informação no banco de dados, a não ser que outra informação não relacionada também seja incluída

Anomalia de exclusão: Pode não ser possível apagar uma informação sem perder outra informação não relacionada

Reengenharia de banco de dados

- Quando projetamos um banco de dados utilizando um processo de modelagem adequado, restarão poucas redundâncias no esquema relacional, mas, ainda sim, podem haver redundâncias que devem ser eliminadas
- Outro processo que gera bancos de dados com informações redundantes é a reengenharia[1]
- No caso da reengenharia, a quantidade de informação redundante pode ser extremamente grande
- A normalização é um processo de modificação que aplica *formas normais* sobre as tabelas de um banco de dados com o intuito de eliminar a redundância de informações

[1] Reengenharia de banco de dados é a obtenção de um novo banco de dados utilizando técnicas de engenharia reversa sobre um banco de dados existente.

Exemplo de banco de dados não normalizado

NP(número do pedido), PE (prazo de entrega), CLI (cliente) , CNPJ (código nacional de pessoa jurídica), E (endereço), CI (cidade) , UF (unidade federativa) , CV (código do vendedor), V (vendedor), CP (código do produto), DP (descrição do produto), UP (unidade do produto) , QP (quantidade do produto), VUP (valor unitário do produto), VT (valor total do produto), TP (total do pedido)

NP	PE	CLI	CNPJ	E	CI	UF	CV	V	CP	DP	UP	QP	VUP	VT	TP
1	10	C	003	cc3	Yyy	YY	25	W	101	ff	UN	2,0	10,0	20,0	232,0
									102	gg	UN	7,0	20,0	140,0	
									103	hh	KG	2,4	30,0	72,0	
2	15	D	004	dd4	Yyy	YY	26	Z	102	gg	UN	6,0	20,0	120,0	931,0
									103	hh	KG	4,3	30,0	129,0	
									104	ii	KG	5,8	40,0	232,0	
									105	jj	L	9,0	50,0	450,0	
3	30	A	001	aa1	Xxx	XX	25	W	101	ff	UN	8,0	10,0	80,0	80,0
4	21	A	001	aa1	Xxx	XX	25	W	102	gg	UN	10,0	20,0	200,0	280,0
									104	ii	KG	2,0	40,0	80,0	
5	10	C	003	cc3	Yyy	YY	26	Z	102	gg	UN	3,0	20,0	60,0	204,0
									104	ii	KG	2,6	40,0	94,0	
									105	jj	L	1,0	50,0	50,0	
6	30	B	002	bb2	Xxx	XX	25	W	104	ii	KG	1,8	40,0	72,0	472,0
									105	jj	L	2,0	50,0	100,0	
									106	kk	L	5,0	60,0	300,0	
7	15	A	001	aa1	Xxx	XX	25	W	102	gg	UN	3,0	20,0	60,0	135,0
									103	hh	KG	2,5	30,0	75,0	

Anomalias no exemplo

- Além do problema de redundância, a tabela anterior apresenta as seguintes anomalias de atualização:

  Anomalia de alteração: Se o preço de um produto for alterado, todas as linhas devem ser varridas para alteração.

  Anomalia de inclusão: Na inclusão de um novo cliente, o mesmo deve estar relacionado a uma venda

  Anomalia de exclusão: Na exclusão de um cliente, os dados dos pedidos relacionados são perdidos

Primeira forma normal (1FN)

- A 1FN visa eliminar colunas multi-valoradas
- No nosso exemplo, o conjunto de colunas relacionado com os produtos são multi-valorados (um pedido possui vários produtos)
- Devido a esse problema, a tabela possui várias linhas com dados duplicados para que um pedido com todos os seus produtos seja armazenado
- Para obtermos tabelas obedecendo a 1FN, devemos decompor a tabela não normalizada em várias tabelas, sendo que, deve existir uma tabela para cada conjunto de colunas multi-valoradas
- As tabelas resultantes da 1FN possuirão como chave primária a chave primária da tabela original mais uma coluna identificadora do conjunto de colunas multi-valoradas

Exemplo com 1FN

- Para o nosso exemplo, teremos as seguintes tabelas obedecendo a 1FN:

```
pedido(numero_pedido, prazo_entrega, cliente, cnpj, endereco, cidade, uf,
    codigo_vendedor, vendedor, total_pedido)
item_pedido(*numero_pedido, codigo_produto, descricao_produto,
    unidade_produto, quantidade_produto, valor_unitario, valor_total)
*item_pedido.numero_pedido: pedido.numero_pedido
```

Tabela pedido (1FN)

```
pedido(numero_pedido, prazo_entrega, cliente, cnpj, endereco, cidade, uf,
    codigo_vendedor, vendedor, total_pedido)
```

NP	PE	Cl	CNP	E	Ci	UF	CV	V	TP
1	10	C	003	cc3	Yyy	YY	25	W	232,0
2	15	D	004	dd4	Yyy	YY	26	Z	931,0
3	30	A	001	aa1	Xxx	XX	25	W	80,0
4	21	A	001	aa1	Xxx	XX	25	W	280,0
5	10	C	003	cc3	Yyy	YY	26	Z	204,0
6	30	B	002	bb2	Xxx	XX	25	W	472,0
7	15	A	001	aa1	Xxx	XX	25	W	135,0

Tabela item de pedido (1FN)

```
item_pedido(
  *numero_pedido, codigo_produto,
  descricao_produto, unidade_produto,
  quantidade_produto,valor_unitario,
  valor_total)
*item_pedido.numero_pedido:
  pedido.numero_pedido
```

NP	CP	DP	UP	QP	VUP	VT
1	101	ff	UN	2,0	10,0	20,0
1	102	gg	UN	7,0	20,0	140,0
1	103	hh	KG	2,4	30,0	72,0
2	102	gg	UN	6,0	20,0	120,0
2	103	hh	KG	4,3	30,0	129,0
2	104	ii	KG	5,8	40,0	232,0
2	105	jj	L	9,0	50,0	450,0
3	101	ff	UN	8,0	10,0	80,0
4	102	gg	UN	10,0	20,0	200,0
4	104	ii	KG	2,0	40,0	80,0
5	102	gg	UN	3,0	20,0	60,0
5	104	ii	KG	2,6	40,0	94,0
5	105	jj	L	1,0	50,0	50,0
6	104	ii	KG	1,8	40,0	72,0
6	105	jj	L	2,0	50,0	100,0
6	106	kk	L	5,0	60,0	300,0
7	102	gg	UN	3,0	20,0	60,0
7	103	hh	KG	2,5	30,0	75,0

Dependência funcional

- Para aplicarmos a 2FN e a 3FN, precisamos entender o conceito de *dependência funcional*
- Um conjunto de colunas não vazio A depende funcionalmente de um conjunto de colunas não vazio B (denotado por $B \rightarrow A$) quando o valor de B determina o valor de A, ou seja, para cada valor do *determinante* B existe um único valor de A
- No nosso exemplo temos $NP \rightarrow PE$, ou seja, **prazo_entrega** depende funcionalmente de **numero_pedido**

Dependência funcional total

- A dependência funcional total ocorre quando uma tabela possui uma chave primária composta (formada por várias colunas) e o valor de uma coluna (que não está na chave primária) depende de todas as colunas da chave primária
- Na tabela **item_pedido** de nosso exemplo temos $NP, CP \rightarrow QP$, isto é, a coluna **quantidade** depende de forma total das colunas **numero_pedido** e **codigo_produto**
- As dependências funcionais totais podem existir sem problemas

Dependência funcional parcial

- A dependência funcional parcial acontece quando o valor de uma coluna depende de parte de uma chave primária composta
- Em nosso exemplo temos $CP \rightarrow UP$, a coluna **unidade** depende parcialmente da coluna **codigo_produto** na tabela **item_pedido**

Dependência funcional transitiva

- Quando uma coluna A depende de uma coluna B que, por sua vez, depende da chave primária C, então A é depende transitivo de C
- Voltando ao nosso exemplo, na tabela **pedido** temos $NP \rightarrow V$, pois $NP \rightarrow CV$ e $CV \rightarrow V$
- Isto significa que **vendedor** depende transitivamente do **número do pedido**

Segunda forma normal (2FN)

- A 2FN consiste em eliminar as dependências funcionais parciais
- Essa eliminação gera novas tabelas para os conjuntos de colunas com dependência funcional parcial
- Cada tabela gerada possui como chave primária as colunas das quais havia a dependência funcional parcial

Exemplo com 2FN

- Considerando a tabela **item_pedido** de nosso exemplo, existe uma dependência funcional parcial que pode ser eliminada gerando a tabela **produto**
- Em nosso exemplo, podemos ter a coluna **valor_unitario** em duas tabelas
- O **valor_unitario** de **produto** armazena o valor atual dos produtos que pode variar com o tempo
- Já o **valor_unitario** de **item_pedido**, armazena o valor em cada pedido

```
pedido(numero_pedido, prazo_entrega, cliente, cnpj, endereco, cidade, uf,
    codigo_vendedor, vendedor, total_pedido)
produto(codigo_produto, descricao_produto, unidade_produto, valor_unitario)
item_pedido(*numero_pedido, *codigo_produto, quantidade_produto, valor_unitario,
    valor_total)
*item_pedido.numero_pedido: pedido.numero_pedido
*item_pedido.codigo_produto: produto.codigo_produto
```

Produto (2FN)

```
produto(codigo_produto, descricao_produto, unidade_produto, valor_unitario)
```

CP	DP	UP	VUP
101	ff	UN	10,0
102	gg	UN	20,0
103	hh	KG	30,0
104	ii	KG	40,0
105	jj	L	50,0
106	kk	L	60,0

Item de pedido (2FN)

```
item_pedido(*numero_pedido, *codigo_produto,
    quantidade_produto, valor_unitario,
    valor_total)
*item_pedido.numero_pedido:
    pedido.numero_pedido
*item_pedido.codigo_produto:
    produto.codigo_produto
```

NP	CP	QP	VUP	VT
1	101	2,0	10,0	20,0
1	102	7,0	20,0	140,0
1	103	2,4	30,0	72,0
2	102	6,0	20,0	120,0
2	103	4,3	30,0	129,0
2	104	5,8	40,0	232,0
2	105	9,0	50,0	450,0
3	101	8,0	10,0	80,0
4	102	10,0	20,0	200,0
4	104	2,0	40,0	80,0
5	102	3,0	20,0	60,0
5	104	2,6	40,0	94,0
5	105	1,0	50,0	50,0
6	104	1,8	40,0	72,0
6	105	2,0	50,0	100,0
6	106	5,0	60,0	300,0
7	102	3,0	20,0	60,0
7	103	2,5	30,0	75,0

Terceira forma normal (3FN)

- A 3FN tem como objetivo eliminar as dependências transitivas
- Na retirada de uma dependência transitiva, cria-se uma nova tabela contendo as colunas com dependência transitiva
- A chave primária da nova tabela será a coluna que *intermediou* a dependência transitiva

Exemplo com 3FN

- Em nosso exemplo, na tabela **pedido**, temos que **vendedor** depende transitivamente de **numero_pedido** por intermédio de **codigo_vendedor**
- Além da dependência transitiva anterior, temos uma dependência transitiva envolvendo as colunas do cliente
- Nesse caso, podemos tomar o **cnpj** como chave primária da tabela a ser criada

```
vendedor(codigo_vendedor, vendedor)
cliente(cnpj, cliente, endereco, cidade, uf)
pedido(numero_pedido, prazo_entrega, *cnpj, *codigo_vendedor,
       total_pedido)
*pedido.cnpj: cliente.cnpj
*pedido.codigo_vendedor: vendedor.codigo_vendedor
```

Pedido, cliente e vendedor na primeira aplicação da 3FN

```
vendedor(codigo_vendedor, vendedor)
```

CV	V
25	Y
26	Z

```
pedido(numero_pedido, prazo_entrega,
    *cnpj, *codigo_vendedor,
    total_pedido)
*pedido.cnpj: cliente.cnpj
*pedido.codigo_vendedor:
    vendedor.codigo_vendedor
```

```
cliente(cnpj, cliente, endereco,
    cidade, uf)
```

CNP	Cl	E	Ci	UF
001	A	aa1	Xxx	XX
002	B	bb2	Xxx	XX
003	C	cc3	Yyy	YY
004	D	dd4	Yyy	YY

NP	PE	CNP	CV	TP
1	10	003	25	232,0
2	15	004	26	931,0
3	30	001	25	80,0
4	21	001	25	280,0
5	10	003	26	204,0
6	30	002	25	472,0
7	15	001	25	135,0

O esquema lógico ainda não está na 3FN

- A tabela **cliente** ainda não está na 3FN, pois **uf** depende transitivamente de **cnpj** por intermédio de **cidade**
- Poderíamos criar a tabela **cidade(cidade, uf)** para resolver esta dependência
- Contudo, se incluirmos a coluna **codigo_cidade** temos uma chave primária melhor do que o nome da cidade

Esquema relacional normalizado

```
vendedor(codigo_vendedor, vendedor)
cidade(codigo_cidade, cidade, uf)
cliente(cnpj, cliente, endereco, *codigo_cidade, uf)
*cliente.codigo_cidade: Cidade.codigo_cidade
pedido(numero_pedido, prazo_entrega, *cnpj, *codigo_vendedor, total_pedido)
*pedido.cnpj: cliente.cnpj
*pedido.codigo_vendedor: Vendedor.codigo_vendedor
produto(codigo_produto, descricao_produto, unidade_produto)
item_pedido(*numero_pedido, *codigo_produto, quantidade_produto,
    valor_unitario, valor_total)
*item_pedido.numero_pedido: pedido.numero_pedido
*item_pedido.codigo_produto: produto.codigo_produto
```

Banco de dados na 3FN: vendedor, cidade, cliente

```
vendedor(codigo_vendedor,
    vendedor)
```

CV	V
25	Y
26	Z

```
cliente(cnpj, cliente, endereco, *codigo_cidade)
```

CNP	Cl	E	CC
001	A	aa1	001
002	B	bb2	001
003	C	cc3	002
004	D	dd4	002

```
cidade(codigo_cidade, cidade,
    uf)
```

CC	Ci	UF
001	Xxx	XX
002	Yyy	YY

Banco de dados na 3FN: produto e pedido

```
produto(codigo_produto,
    descricao_produto, unidade_produto,
    valor_unitario)
```

CP	DP	UP	VUP
101	ff	UN	10,0
102	gg	UN	20,0
103	hh	KG	30,0
104	ii	KG	40,0
105	jj	L	50,0
106	kk	L	60,0

```
pedido(numero_pedido, prazo_entrega,
    *cnpj, *codigo_vendedor,
    total_pedido)
```

NP	PE	CNP	CV	TP
1	10	003	25	232,0
2	15	004	26	931,0
3	30	001	25	80,0
4	21	001	25	280,0
5	10	003	26	204,0
6	30	002	25	472,0
7	15	001	25	135,0

Banco de dados na 3FN: item de pedido

```
item_pedido(*numero_pedido, *codigo_produto,
    quantidade_produto, valor_unitario,
    valor_total)
```

NP	CP	QP	VUP	VT
1	101	2,0	10,0	20,0
1	102	7,0	20,0	140,0
1	103	2,4	30,0	72,0
2	102	6,0	20,0	120,0
2	103	4,3	30,0	129,0
2	104	5,8	40,0	232,0
2	105	9,0	50,0	450,0
3	101	8,0	10,0	80,0
4	102	10,0	20,0	200,0
4	104	2,0	40,0	80,0
5	102	3,0	20,0	60,0
5	104	2,6	40,0	94,0
5	105	1,0	50,0	50,0
6	104	1,8	40,0	72,0
6	105	2,0	50,0	100,0
6	106	5,0	60,0	300,0
7	102	3,0	20,0	60,0
7	103	2,5	30,0	75,0

Referências

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 6. ed. São Paulo: Pearson Addison Wesley, 2011.

HEUSER, C. A. **Projeto de banco de dados**. 6. ed. Porto Alegre: Bookman, 2009.

RAMAKRISHNAN, R.; GEHRKE, J. **Sistemas de gerenciamento de banco de dados**. 3. ed. São Paulo: McGrawHill, 2008.