Aula 015

Professores:

Geraldo Xexéo

Geraldo Zimbrão

Conteúdo:

Formas Normais

Fundação CECIERJ
Consórcio cederj

Relembrando

- Tabelas (Relações)
- Linhas (Tuplas)
- Colunas (Atributos)
- Chaves
  - Chaves Primárias
  - Chaves Estrangeiras

Tabelas? Quais Tabelas?

- Em um banco de dados relacional, devemos **organizar** nossos dados em tabelas.
- Mas como escolher essa organização? Que tabelas e que atributos escolher?
- Por que isso é um problema?

O Tabelão

- Muitas pessoas pensam em colocar todos os seus dados em uma só tabela.

- Vamos ver um exemplo
  - Fornecedores (número, status, cidades)
  - Peças que entregam (número)
  - Quantidade

TABELÃO				
NUMF	STATUS	CIDADE	NUMP	QUANTIDADE
1	20	São Paulo	1	300
1	20	São Paulo	2	200
1	20	São Paulo	3	400
1	20	São Paulo	4	200
1	20	São Paulo	5	100
1	20	São Paulo	6	100
2	10	Rio de Janeiro	1	300
2	10	Rio de Janeiro	2	400
3	10	Rio de Janeiro	2	200
4	20	São Paulo	2	200
4	20	São Paulo	4	300
4	20	São Paulo	5	400

Problemas do Tabelão (1)

- Ao inserir
  - Não podemos guardar a informação que um fornecedor está em uma cidade se ele não fornece uma peça
    - A chave é NUMP+NUMF

TABELÃO				
NUMF	STATUS	CIDADE	NUMP	QUANTIDADE
1	20	São Paulo	1	300
1	20	São Paulo	2	200
1	20	São Paulo	3	400
1	20	São Paulo	4	200
1	20	São Paulo	5	100
1	20	São Paulo	6	100
2	10	Rio de Janeiro	1	300
2	10	Rio de Janeiro	2	400
3	10	Rio de Janeiro	2	200
4	20	São Paulo	2	200
4	20	São Paulo	4	300
4	20	São Paulo	5	400

Problemas do Tabelão (2)

- Ao apagar
  - Se apagamos a única tupla de um pedido para o fornecedor (por exemplo, para o fornecedor 3), deixamos de saber outros dados sobre ele, como seu status e sua localização

Problemas do Tabelão (3)

- Ao alterar
  - Se queremos alterar o status de um fornecedor que aparece em muitas linhas, temos que ter o cuidado, e o trabalho, de alterar todas as linhas

Qual a Solução?

- Alterar o projeto para evitar os problemas
- Por exemplo, dividindo a tabela em duas

Solução

TABELÃO				
NUMF	STATUS	CIDADE	NUMP	QUANTIDADE
1	20	São Paulo	1	300
1	20	São Paulo	2	200
1	20	São Paulo	3	400
1	20	São Paulo	4	200
1	20	São Paulo	5	100
1	20	São Paulo	6	100
2	10	Rio de Janeiro	1	300
2	10	Rio de Janeiro	2	400
3	10	Rio de Janeiro	2	200
4	20	São Paulo	2	200
4	20	São Paulo	4	300
4	20	São Paulo	5	400

FORNECEDOR		
NUMF	STATUS	CIDADE
1	20	São Paulo
2	10	Rio de Janeiro
3	10	Rio de Janeiro
4	20	São Paulo
5	30	Belo Horizonte

PEDIDOS		
NUMF	NUMP	QUANTIDADE
1	1	300
1	2	200
1	3	400
1	4	200
1	5	100
1	6	100
2	1	300
2	2	400
3	2	200
4	2	200
4	4	300
4	5	400

Mas... Como fazer?

- Problemas desse tipo podem ocorrer em qualquer projeto, de forma mais ou menos explícita
  - Conhecidos como **anomalias**
- Os pesquisadores buscaram uma maneira de produzir regras gerais que criassem, a partir de um projeto com defeitos, um projeto correto
  - Imune a problemas desse tipo

Normalização

- Decomposição de esquemas para evitar as anomalias
  - As anomalias são fruto da redundância de dados

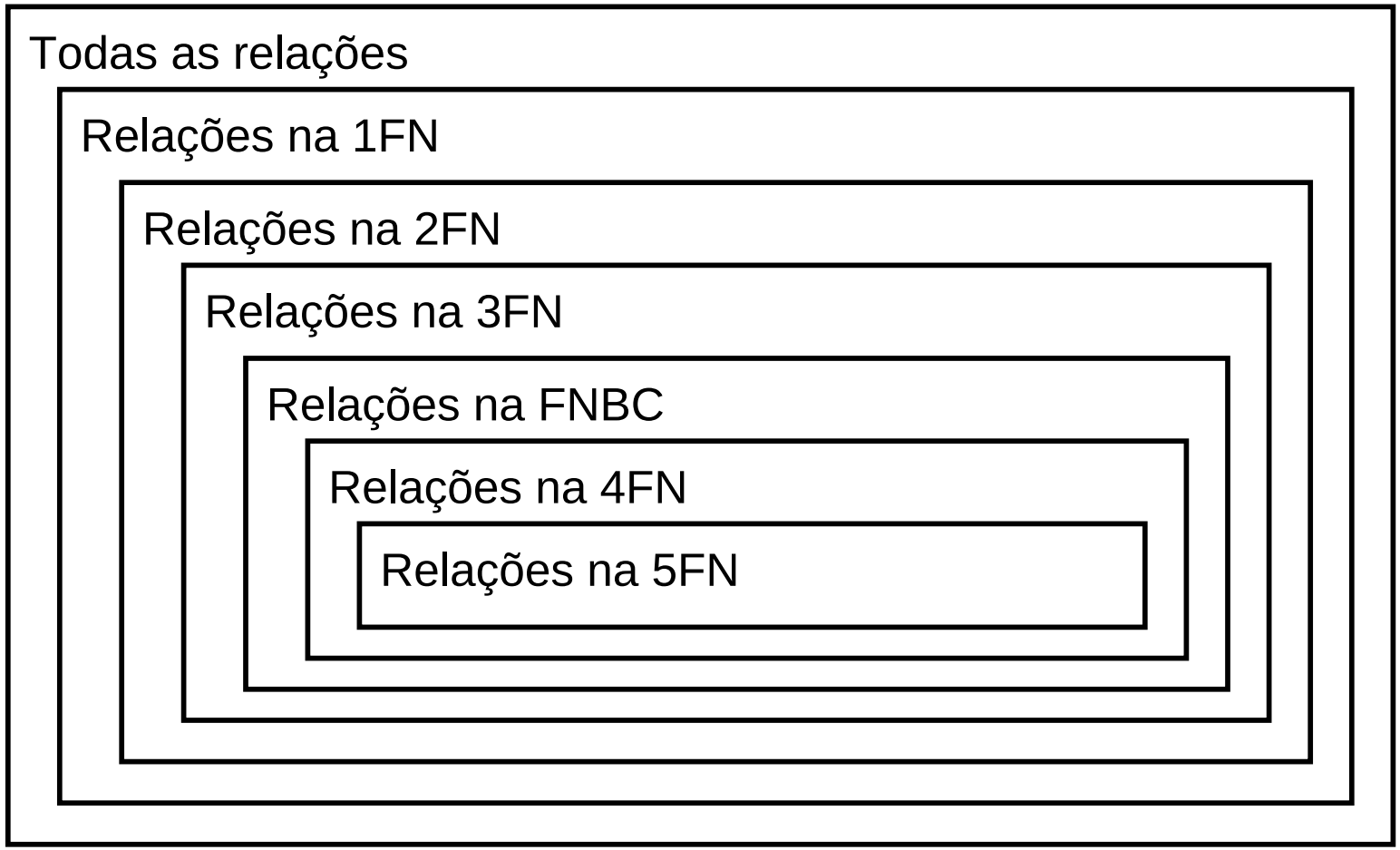
- Mecanismo formal de análise de esquemas baseados na chave e nas *dependências funcionais*

Normalização

- Os esquemas resultantes, normalizados, devem possuir características, que são analisadas em ordem de complexidade

- Formas Normais
  - Primeira (1FN), Segunda (2FN), Terceira (3FN), Boyce-Codd (FNBC), Quarta (4FN), Quinta (5FN)

Hierarquia das FN



Até onde ir?

- Teoricamente, até a 5FN
- Na prática, temos a 1FN e nos preocupamos razoavelmente até 3FN
- Na implementação, muitas vezes desnormalizamos!
  - Mas isso fica para outro curso

Nesse curso

- Veremos até a 3FN + BCFN
- A 4FN e a 5FN ficam para o curso de Banco de Dados
  - Mais complexas
  - Menos utilizadas
- Porém, você pode pesquisá-las se quiser

Observação

- Um projeto conceitual bem feito já evita as anomalias
  - Na prática, os mesmos princípios que veremos aqui devem estar presentes nos seus modelos de entidades e relacionamentos

Como começar?

- Para partir do "Tabelão" para tabelas menores onde não existem anomalias, diminuindo a redundância, temos que:
  - Decompor a relação
  - Não perder informação

Decomposição e Perdas

- Vamos ver um exemplo muito simples de perda de informação na decomposição

FORNECEDOR		
NUMF	STATUS	CIDADE
3	30	Rio de Janeiro
5	30	Belo Horizonte

F1	
NUMF	STATUS
3	30
5	30

F2	
STATUS	CIDADE
30	Rio de Janeiro
30	Belo Horizonte

Qual o problema?

Que fornecedor está em que cidade?

F1	
NUMF	**STATUS**
3	30
5	30

F2	
STATUS	**CIDADE**
30	Rio de Janeiro
30	Belo Horizonte

Qual a Decomposição Correta?

F1	
NUMF	**CIDADE**
3	Rio de Janeiro
5	Belo Horizonte

F2	
NUMF	**STATUS**
3	30
5	30

- A decomposição correta é aquela que permite que a relação original seja recriada com um JOIN

- Decomposição sem perda

Join?

- FORNECEDOR = F1 JOIN F2
  - Campo comum: NUMF

F1	
NUMF	**CIDADE**
3	Rio de Janeiro
5	Belo Horizonte

⋈

F2	
NUMF	**STATUS**
3	30
5	30

FORNECEDOR		
NUMF	**STATUS**	**CIDADE**
3	30	Rio de Janeiro
5	30	Belo Horizonte

Projeções

FORNECEDOR		
NUMF	STATUS	CIDADE
3	30	Rio de Janeiro
5	30	Belo Horizonte

- Decomposições são projeções

- F1 = FORNECEDOR [NUMF, CIDADE]

F1	
NUMF	CIDADE
3	Rio de Janeiro
5	Belo Horizonte

- F2 = FORNECEDOR [NUMF, STATUS]

F2	
NUMF	STATUS
3	30
5	30

Pergunta

- Se R1 e R2 são projeções de uma relação R, e R1 e R2 incluem entre si todos os atributos de R, quais condições devem ser satisfeitas para que o JOIN de R1 e R2 recupere R

Dependência Funcional

- Se R é uma relação e X e Y são subconjuntos arbitrários do conjunto de atributos de R, dizemos que Y é funcionalmente dependente de X se e somente se para todos os possíveis valores legais de R caso duas tuplas de R tenham os mesmos valores para os atributos de X então terão os mesmos valores para os atributos de Y
  - com a notação

$$X \rightarrow Y$$

Sendo mais claro

- Y ser dependente funcional de X significa que quando definimos o valor de X, o valor de Y fica automaticamente definido

- Y é função de X

DF: Exemplo (1)

- FUNCIONÁRIO = { NOME, SALÁRIO, SEXO }
  - { NOME } → { SALÁRIO }
  - { NOME } → { SEXO }
  - { NOME } → { SALÁRIO , SEXO }
  - { NOME } → { NOME SALÁRIO , SEXO }
  - { NOME, SALÁRIO } → { SALÁRIO , SEXO }

DF: Exemplo (2)

TABELÃO				
NUMF	STATUS	CIDADE	NUMP	QUANTIDADE
1	20	São Paulo	1	300
1	20	São Paulo	2	200
1	20	São Paulo	3	400
1	20	São Paulo	4	200
1	20	São Paulo	5	100
1	20	São Paulo	6	100
2	10	Rio de Janeiro	1	300
2	10	Rio de Janeiro	2	400
3	10	Rio de Janeiro	2	200
4	20	São Paulo	2	200
4	20	São Paulo	4	300
4	20	São Paulo	5	400

No tabelão

- { NUMF } → { STATUS }
- { NUMF } → { CIDADE }
- { NUMF, NUMP } → { QUANTIDADE }
- { NUMF, NUMP } → { CIDADE }
- { NUMF, NUMP } → { CIDADE, QUANTIDADE }
- { NUMF, NUMP } → {NUMF, NUMP, CIDADE, QUANTIDADE
- E outras

DF: Exemplo (3)

- Porém, algumas DF não são válidas no tabelão
  - FALSO: { NUMF } → { NUMP }
  - FALSO: { NUMP } → { CIDADE }
  - FALSO: { NUMP } → { STATUS }
  - E outros

Regras para as DF (1)

- Reflexividade:
  - Se A ⊃ B, A → B
- Aumento:
  - Se A → B então AC → BC
- Transitividade:
  - Se A → B e B → C então A → C

Regras para as DF (2)

- Auto-determinação:
  - $A \rightarrow A$
- Decomposição:
  - Se $A \rightarrow BC$ então $A \rightarrow B$ e $A \rightarrow C$
- União:
  - Se $A \rightarrow B$ e $A \rightarrow C$ então $A \rightarrow BC$

Regras para as DF (3)

- Composição
  - Se A → B e C → D então AC → BD

DF: Nomes

- O determinante é o lado esquerdo da equação de dependência funcional

- O dependente é o lado direito da equação de dependência funcional

Observação

- A dependência funcional é uma questão semântica
  - Depende dos dados e não apenas do esquema da relação

Primeira Forma Normal

- Uma relação está na Primeira Forma Normal (1FN) se e somente se todos os seus domínios só possuem valores escalares
  - Todos os seus atributos só possuem valores atômicos

NF^2

- Non First Normal Formal
- Fora da Primeira (e de todas) forma normal

FORNECEDOR			
NUMF	STATUS	CIDADE	PEÇAS
1	20	São Paulo	1,2,3,4
2	10	Rio de Janeiro	2,4,5
3	10	Rio de Janeiro	1,2,6
4	20	São Paulo	1,2,5
5	30	Belo Horizonte	2.6

NÃO ESTÁ NA 1FN

Segunda Forma Normal

- Uma relação está na segunda forma normal se e somente se ela está na 1FN e todos os seus atributos que não pertencem a chave são dependentes funcionalmente de toda a chave e não de um subconjunto da chave
  - Se a chave só tem um atributo, automaticamente a tabela está na 2FN

2FN: Exemplo (1)

- A relação TABELÃO está na 1FN e não está na 2FN
  - Chave = { NUMF, NUMP }
  - { NUMF } → { STATUS }

TABELÃO				
NUMF	STATUS	CIDADE	NUMP	QUANTIDADE
1	20	São Paulo	1	300
1	20	São Paulo	2	200
1	20	São Paulo	3	400
1	20	São Paulo	4	200
1	20	São Paulo	5	100
1	20	São Paulo	6	100
2	10	Rio de Janeiro	1	300
2	10	Rio de Janeiro	2	400
3	10	Rio de Janeiro	2	200
4	20	São Paulo	2	200
4	20	São Paulo	4	300
4	20	São Paulo	5	400

NÃO ESTÁ NA 2FN

2FN: Exemplo 2

- A relação FORNECEDOR está na 2FN
  - A chave só contém um atributo

FORNECEDOR		
NUMF	**STATUS**	**CIDADE**
1	20	São Paulo
2	10	Rio de Janeiro
3	10	Rio de Janeiro
4	20	São Paulo
5	30	Belo Horizonte

2FN: Exemplo 3 (1)

- A relação PEDIDO está na 2FN
  - Chave = { NUMF, NUMP }
  - { NUMF, NUMP } ⟼ { QUANTIDADE }
  - É Falso que { NUMP } ⟼ { QUANTIDADE }
  - É Falso que { NUMF } ⟼ { QUANTIDADE }

PEDIDO		
NUMF	**NUMP**	**QUANTIDADE**
1	1	300
1	2	200
1	3	400
1	4	200
1	5	100
1	6	100
2	1	300
2	2	400
3	2	200
4	2	200
4	4	300
4	5	400

2FN: Exemplo 3 (2)

PEDIDO		
NUMF	NUMP	QUANTIDADE
1	1	300
1	2	200
1	3	400
1	4	200
1	5	100
1	6	100
2	1	300
2	2	400
3	2	200
4	2	200
4	4	300
4	5	400

- Chave = { NUMF, NUMP }
- { NUMF, NUMP } → { QUANTIDADE }
- É Falso que { NUMP } → { QUANTIDADE }
- É Falso que { NUMF } → { QUANTIDADE }

Terceira Forma Normal

- Uma relação está na 3FN quando está na 2FN e todo atributo não-chave é dependente funcional apenas da chave e não é dependente funcional de nenhum outro atributo fora da chave
  - Não é transitivamente dependente

3FN: Exemplo (1)

NÃO ESTÁ NA 3FN

- PESSOA = { NOME, NOMEPAI, NOMEAVÔ }
- CHAVE = { NOME } garante 2FN
- { NOMEPAI } → { NOMEAVÔ }
  - A relação não está na 3FN

3FN: Exemplo 2

AUTOMÓVEL			
PLACA	MODELO	COR	FÁBRICA
KNH-4440	GOL	AZUL	VOLKSWAGEM
KSD-2002	PARATI	VERDE	VOLKSWAGEM
LAS-2185	GOL	AMARELO	VOLKSWAGEM
TTT-0007	SIENA	VERDE	FIAT

NÃO ESTÁ NA 3FN

- CHAVE = { PLACA }
- { PLACA } → { MODELO }
- { PLACA } → { FÁBRICA }
- { MODELO } → { FÁBRICA }

3FN: Exemplo 3

PEDIDO			
NUMF	**NUMP**	**QUANTIDADE**	**PRAZO**
1	1	300	04/10/2006
1	2	200	06/03/2006
1	3	400	07/02/2006
1	4	200	10/01/2006
1	5	100	10/11/2006
1	6	100	10/16/2006
2	1	300	11/29/2006
2	2	400	12/26/2006
3	2	200	03/07/2007
4	2	200	03/24/2007
4	4	300	06/03/2007
4	5	400	08/30/2007

- Esta relação está na 3FN
- Ambos, prazo e quantidade dependem da chave e apenas da chave completa
- Não dependem um do outro

BCNF

- Forma Normal Boyce-Codd
- Uma melhoria na 3FN
  - Casos onde há mais de uma chave candidata
  - Chaves candidatas compostas
  - Com atributo em comum
- Se isso não ocorre, é igual a 3FN

BCNF

- Uma relação está na BCNF se e somente se os únicos determinantes são as chaves candidatas
  - Ou seja, dadas todas as equações de DF válidas para a relação, em todas elas os determinantes são chaves candidatas

3FN: Exemplo 1

AUTOMÓVEL			
PLACA	MODELO	COR	FÁBRICA
KNH-4440	GOL	AZUL	VOLKSWAGEM
KSD-2002	PARATI	VERDE	VOLKSWAGEM
LAS-2185	GOL	AMARELO	VOLKSWAGEM
TTT-0007	SIENA	VERDE	FIAT

NÃO ESTÁ NA BCNF

- CHAVE = { PLACA }
- { PLACA } → { MODELO }
- { PLACA } → { FÁBRICA }
- { MODELO } → { FÁBRICA }
- { MODELO } NÃO É CHAVE CANDIDATA

3FN: Exemplo 2

- CHAVE = { PLACA }
- { PLACA } → { MODELO }
- { PLACA } → { COR }
- + as triviais
- Todos determinantes são chaves candidatas

AUTOMÓVEL		
PLACA	MODELO	COR
KNH-4440	GOL	AZUL
KSD-2002	PARATI	VERDE
LAS-2185	GOL	AMARELO
TTT-0007	SIENA	VERDE

Conclusão

- Devemos evitar anomalias em nossos projetos de bancos de dados
- Devemos buscar as formas normais
- Podemos e devemos fazer isso já no projeto conceitual