Normalização de dados

Origem: Wikipédia, a enciclopédia livre.

Normalização de banco de dados é um conjunto de regras que visa, principalmente, a organização de um projeto de banco de dados para reduzir a redundância de dados, aumentar a integridade de dados e o desempenho. Para normalizar o banco de dados, deve-se examinar as colunas (atributos) de uma entidade e as relações entre entidades (tabelas), com o objetivo de se evitar anomalias observadas na inclusão, exclusão e alteração de registros.

Atualmente, muitos sistemas de informação ainda não utilizam banco de dados relacionais, sendo esses chamados de sistemas legados. Os dados desses sistemas são armazenados em arquivos de linguagens de terceira geração, como COBOL ou BASIC, ou então, em banco de dados da era pré-relacional.

Índice

Panorâmica informal

Para adequar o banco de dados, é necessário avaliar com base em cinco regras, que recebem o nome de formas normais. Essas correspondem a um conjunto de regras de simplificação e adequação de tabelas. Diz-se que a tabela do banco de dados relacional está numa certa forma normal quando satisfaz as condições exigentes. O trabalho original de Edgar F. Codd, definiu três dessas formas, mas existem hoje outras formas normais geralmente aceitas. Damos aqui uma curta panorâmica informal das mais comuns. Cada forma normal listada abaixo representa uma condição mais forte das que a precedem na lista. Para a maioria dos efeitos práticos, considera-se que as bases de dados estão normalizadas se aderirem à terceira forma normal.

Inicialmente, são definidos todos os atributos do documento, que estão relacionados a uma entidade principal, atribuindo uma chave primária. Feito isso, partimos para a análise do documento de acordo com as formas normais a seguir:

- **Primeira Forma Normal** (ou **1FN**). Nesta forma os atributos precisam ser atômicos, o que significa que as tabelas não podem ter valores repetidos e nem atributos possuindo mais de um valor. Exemplo: CLIENTE = {ID + ENDEREÇO + *TELEFONES*}. Porém, uma pessoa poderá ter mais de um número de telefone, sendo assim o atributo "TELEFONES" é multivalorado. Para normalizar, é necessário:
  1. Identificar a chave primária e também a coluna que possui dados repetidos (nesse exemplo "TELEFONES") e removê-los;
  2. Construir uma outra tabela com o atributo em questão, no caso "TELEFONES". Mas não se esquecendo de fazer uma relação entre as duas tabelas: CLIENTE = {ID + ENDEREÇO} e TELEFONE (nova tabela) = {CLIENTE_ID (chave

estrangeira) + TELEFONE}.

- **Segunda Forma Normal** (ou **2FN**). Primeiramente, para estar na **2FN** é preciso estar também na **1FN**. 2FN define que os atributos normais, ou seja, os não chave, devem depender unicamente da chave primária da tabela. Assim como as colunas da tabela que não são dependentes dessa chave devem ser removidas da tabela principal e cria-se uma nova tabela utilizando esses dados. Exemplo: PROFESSOR_CURSO = {ID_PROF + ID_CURSO + SALARIO + *DESCRICAO_CURSO*} Como podemos observar, o atributo "DESCRICAO_CURSO" não depende unicamente da chave primária "ID_PROF", mas sim somente da chave "ID_CURSO". Para normalizar, é necessário:

  1. Identificar os dados não dependentes da chave primária (nesse exemplo "DESCRICAO_CURSO") e removê-los;
  2. Construir uma nova tabela com os dados em questão: PROFESSOR_CURSO = {ID_PROF + ID_CURSO + SALARIO} e CURSOS (nova tabela) = {ID_CURSO + DESCRICAO_CURSO}.

- **Terceira Forma Normal** (ou **3FN**). Assim como para estar na **2FN** é preciso estar na **1FN**, para estar na **3FN** é preciso estar também na **2FN**. 3FN define que todos os atributos dessa tabela devem ser funcionalmente independentes uns dos outros, ao mesmo tempo que devem ser dependentes exclusivamente da chave primária da tabela. 3NF foi projetada para melhorar o desempenho de processamento dos banco de dados e minimizar os custos de armazenamento. Exemplo: FUNCIONARIO = {ID + NOME + VALOR_SALARIO + *VALOR_FGTS*}. Como sabemos o valor do FGTS é proporcional ao salário, logo o atributo normal "VALOR_FGTS" é dependente do também atributo normal "VALOR_SALARIO". Para normalizar, é necessário:

  1. Identificar os dados dependentes de outros (nesse exemplo "VALOR_FGTS");
  2. Removê-los da tabela. Esses atributos poderiam ser definitivamente excluídos -- e deixando para a camada de negócio a responsabilidade pelo seu cálculo -- ou até ser movidos para uma nova tabela e referenciar a principal ("FUNCIONARIO").

- **Forma Normal de *Boyce-Codd*** (ou **BCNF**) requer que não exista nenhuma dependência funcional não trivial de atributos em algo mais do que um superconjunto de uma chave candidata. Neste estágio, todos os atributos são dependentes de uma chave, de uma chave inteira e de nada mais que uma chave (excluindo dependências triviais, como A → A);
- **Quarta Forma Normal** (ou **4FN**) requer que não exista nenhuma dependência multi-valorada não trivial de conjuntos de atributo em algo mais de que um superconjunto de uma chave candidata;
- **Quinta Forma Normal** (ou **5FN** ou **PJ/NF**) requer que não exista dependências de *joins* (associações) não triviais que não venham de restrições chave;
- ***Domain-Key Normal Form*** (ou **DK/NF**) requer que todas as restrições sigam os domínios e restrições chave.

Visão Formal

Antes de falar sobre normalização, é necessário utilizar alguns termos a partir do modelo relacional e defini-los na teoria de conjuntos. Estas definições muitas vezes serão simplificações de seus significados originais, uma vez que somente alguns aspectos do modelo relacional são levados em consideração na normalização.

As notações básicas utilizadas no modelo relacional são *nomes de relacionamentos* e *nomes de atributos*, representados por cadeias de caracteres tais como *Pessoas* e *Nomes*; geralmente são utilizadas variáveis como *r*, *s*, *t*,... e *a*, *b*, *c* para o conjunto dados definido sobre eles. Outra notação básica é o conjunto de *valores atômicos* que contém valores tais como números e cadeias de caracteres.

A primeira definição de interesse é a noção de *tupla*, que formaliza a noção de linha ou registro em uma tabela:

> **Def.** Uma *tupla* é uma função parcial de nomes de atributos para valores atômicos.
> **Def.** Um *cabeçalho* é um conjunto finito de nomes de atributos.
> **Def.** A *projeção* de uma tupla *t* em um conjunto finito de atributos *A* é $t[A] = \{ (a, v): (a, v) \in t, a \in A \}$.

A próxima definição é a de *relação* na qual formaliza-se o teor de uma tabela como ele é definido no modelo relacional.

> **Def.** Uma *relação* é uma tupla (*H*, *B*) sendo *H* um cabeçalho e *B* o corpo, um conjunto de tuplas em que possuem todas o domínio *H*.

Como uma relação corresponde definitivamente com aquela que é usualmente chamada de extensão de um predicado em lógica de primeira ordem exceto que aqui nós identificamos os locais no predicado com nomes de atributos. Geralmente no modelo relacional um esquema de banco de dados é dito consistir-se de um conjunto de nomes relação, os cabeçalhos que são associados com esses nomes e as restrições que devem manter toda instância do esquema de banco de dados. Para normalização nós nos concentraremos nas restrições que indicam relações individuais, isto é, as *restrições relacionais*. O propósito destas restrições é descrever o *universo relacional*, ou seja, o conjunto de todas as relações que são permitidas para serem associadas com certos nomes de relação.

Def. Um *universo relacional U* sobre um cabeçalho *H* é um conjunto não vazio de relações com o cabeçalho *H*.
Def. Um *esquema relacional* (*H*, *C*) consiste de um cabeçalho *H* e um predicado *C*(*R*) que é definido por todas as relações *R* com o cabeçalho *H*.
Def. Uma relação satisfaz o esquema relacional (*H*, *C*) se possuir o cabeçalho *H* e satisfizer *C*.

Restrições Chave e Dependências Funcionais

A restrição relacional mais importante é a restrição de *Chave*.

Ela relaciona cada registro (tupla) a um (ou mais) valor índice.

Def. Uma *Chave* é um atributo que identifica um registro (tupla).

Objetivos de normalização

Um objetivo básico da primeira forma normal, definida por Codd em 1970, era permitir dados serem questionados e manipulados usando uma "sub-linguagem de dados universal" atrelada à lógica de primeira ordem. Questionando e manipulando dados em uma estrutura de dados não normalizada, como a seguinte representação não-1NF de transações de clientes de cartão de crédito, envolve mais complexidade do que seria realmente necessário:

Transações de clientes de cartão de crédito, não normalizada

<table>
<tr><th>Cliente</th><th>Cliente ID</th><th>Transação</th></tr>
<tr><td>João</td><td>1</td><td>
<table>
<tr><th>Tr. ID</th><th>Data</th><th>Valor</th></tr>
<tr><td>12890</td><td>14/out/2003</td><td>-87</td></tr>
<tr><td>12904</td><td>15/out/2003</td><td>-50</td></tr>
</table>
</td></tr>
<tr><td>Wilson</td><td>2</td><td>
<table>
<tr><th>Tr. ID</th><th>Data</th><th>Valor</th></tr>
<tr><td>12898</td><td>14/out/2003</td><td>-21</td></tr>
</table>
</td></tr>
<tr><td>Márcio</td><td>3</td><td>
<table>
<tr><th>Tr. ID</th><th>Data</th><th>Valor</th></tr>
<tr><td>12907</td><td>15/out/2003</td><td>-18</td></tr>
<tr><td>14920</td><td>20/nov/2003</td><td>-70</td></tr>
<tr><td>15003</td><td>27/nov/2003</td><td>-60</td></tr>
</table>
</td></tr>
</table>

Note que a tabela é composta pelas colunas cliente e transação, sendo que essa última contém 3 informações: identificador, data e valor.

Para cada cliente corresponde um grupo repetitivo de transações.

A análise automatizada de transação envolve dois estágios:

1. Desempacotar um ou mais grupos de clientes de transações permitindo transações individuais serem agrupadas para exame, e
2. Derivar o resultado de uma consulta em resultados do primeiro estágio.

Por exemplo, para encontrar a soma monetária de todas as transações que ocorreram em outubro de 2003 para todos os clientes, o sistema necessitaria saber primeiro que precisa desempacotar o grupo de transações para cada cliente, então somar a quantidade de todas as transações de outubro de 2003.

Um das visões mais importantes de Codd foi que a complexidade desta estrutura poderia ser removida completamente, levando a um poder e flexibilidade muito maior na forma de efetuar consultas. A normalização equivalente da estrutura acima seria assim:

Tabela dos clientes

Cliente	Cliente ID
João	1
Wilson	2
Márcio	3

Tabela das transações

Cliente ID	Tr. ID	Data	Valor
1	12890	14/out/2003	-87
1	12904	15/out/2003	-50
2	12898	14/out//2003	-21
3	12907	15/out/2003	-18
3	14920	20/nov/2003	-70
3	15003	27/nov/2003	-60

Nessa nova estrutura, as chaves são {Cliente} e {Cliente ID} na primeira tabela e {Cliente ID, Tr. ID} na segunda.

Agora cada linha representa uma transação individual, e um SGBD pode obter a resposta, simplesmente encontrando todas as linhas com data de outubro, somando então os valores.

Formas Normais

Primeira Forma Normal

Definição pelo Osório

> '*Uma tabela está na 1FN, se e somente se, todos os valores das colunas da tabela forem* ***atómicos'***

> Assim, podemos dizer que os relacionamentos, como definidos acima, estão necessariamente na 1FN. Uma relação está na 1FN quando todos os atributos da relação estiverem baseados em um domínio simples, não contendo grupos ou valores repetidos.

Passagem à 1FN

1. Encontre a chave primária da tabela;
2. Fique ciente de quais são as colunas da tabela que apresentam dados repetidos para que sejam removidas;
3. Crie uma tabela para esses dados repetidos, com a chave primária da anterior;
4. Por fim, estabeleça relação entre a nova tabela e a principal.

Outra forma de identificar se a tabela não está na *1FN* é verificando se existe tabela aninhadas, ou seja, mais de um registro para uma chave primária.

Exemplo

Observe a seguinte tabela:

PEDIDOS = {COD_PEDIDO + CLIENTE + VENDEDOR + ATENDENTE + PRODUTO + QUANT + VALOR}

Considere um pedido como o número **00001**, para este se observarmos o formulário em papel teremos muitos campos a considerar, contudo usaremos apenas alguns para facilitar o entendimento.

Neste momento devemos idealizar o pedido número **00001** e efetuar os seguintes testes:

{COD_PEDIDO | CLIENTE | VENDEDOR | ATENDENTE | PRODUTO | QUANT | VALOR}

{**00001** | "**ANDRÉ**" | "**MARCO**" | "**JOAO**" | "TENIS " | "1" | "50.00"}

{**00001** | "**ANDRÉ**" | "**MARCO**" | "**JOAO**" | "SANDALIA" | "2" | "80.00"}

{**00001** | "**ANDRÉ**" | "**MARCO**" | "**JOAO**" | "CARTEIRA" | "1" | "35.00"}

Observe que para os dados do pedido **00001** lançados acima, apenas os atributos que estão em negrito SÃO ÚNICOS, pois não se diferem. Os demais atributos mudam, não cumprindo a *1FN* onde os atributos devem ser atômicos, quer dizer únicos.

Para testarmos um dos atributos e ter certeza que este é atômico, podemos efetuar uma pergunta conforme o exemplo abaixo:

Podemos ter outro cliente para o pedido **00001**? Não. Podemos ter apenas 1 cliente por pedido, sendo assim este atributo é atômico único para 1 pedido.

Podemos ter outro vendedor para o pedido **00001**? Não. Podemos ter apenas 1 vendedor por pedido.

Podemos ter outro produto para o pedido **00001**? Sim. Podemos ter vários produtos para um pedido, sendo assim, os campos aninhados devem ser extraídos para outra tabela.

Problemas

- Redundância;
- Anomalias de Atual.

Segunda Forma Normal

Definição pelo Osório

> **Uma relação está na 2FN se, e somente se, estiver na 1FN e cada atributo não-chave for dependente da chave primária inteira, isto é, cada atributo não-chave não poderá ser dependente de apenas parte da chave.**

No caso de tabelas com chave primária composta, se um atributo depende apenas de uma parte da chave primária, então esse atributo deve ser colocado em outra tabela.

Passagem à 2FN

1. Geração de novas tabelas com DFs (Dependências Funcionais) completas.
2. Análise de dependências funcionais:
   - *tipo* e *descrição* dependem de *codp*;
   - *nome*, *categ* e *salário* dependem de *code*;
   - *data_início* e *tempo_aloc* dependem de toda a chave.

Conclusões

- Maior independência de dados;
- Redundâncias e anomalias: dependências funcionais indirectas.

Terceira Forma Normal

Definição pelo Osório

> Uma relação *R* está na 3FN se ela estiver na 2FN e cada atributo não-chave de *R* não possuir dependência transitiva, para cada chave candidata de *R*. Todos os atributos dessa tabela devem ser **independentes** uns dos outros, ao mesmo tempo que devem ser **dependentes** exclusivamente da chave primária da tabela.

Exemplo ilustrativo

A tabela a seguir **não está na Terceira Forma Normal** porque a coluna *Total* é dependente, ou é resultado, da multiplicação das colunas *Preço* e *Quantidade*, ou seja, a coluna total tem dependência transitiva de colunas que não fazem parte da chave primária, ou mesmo candidata da tabela. Para que essa tabela passe à Terceira FN o campo *Total* deverá ser eliminado, a fim de que nenhuma coluna tenha dependência de qualquer outra que não seja exclusivamente chave.

Itens do pedido				
Pedido	**Item**	**Preço**	**Quantidade**	**Total**
15	102	9,25	2	18,5
15	132	1,3	5	6,5

Aqui, após o atributo/coluna *Total* ser excluído da tabela, ela já na 3ª Forma Normal. Esse atributo pode ser movido para outra tabela referenciando a antiga.

Itens do pedido			
Pedido	**Item**	**Preço**	**Quantidade**
15	102	9,25	2
15	132	1,3	5

Passagem à 3FN

- Para estar na 3FN precisa estar na 2FN;
- Geração de novas tabelas com DF diretas;
- Análise de dependências funcionais entre atributos não chave;
- Verificar a dependência exclusiva da chave primária;
- Entidades na 3FN também não podem conter atributos que sejam resultados de algum cálculo de outro atributo.

Conclusões

- Maior independência de dados;
- 3FN gera representações lógicas finais na maioria das vezes;
- Redundâncias e anomalias: dependências funcionais.

Quarta Forma Normal

Definição

> **Uma tabela está na 4FN, se e somente se, estiver na 3FN e não existirem dependências multivaloradas.**

Exemplo (base de dados sobre livros)

```
Relação não normalizada: Livros(nrol, (autor), título, (assunto), editora, cid_edit, ano_public)
```

```
1FN: Livros(nrol, autor, assunto, título, editora, cid_edit, ano_public)
```

```
2FN: Livros(nrol, título, editora, cid_edit, ano_public)
     AutAssLiv(nrol, autor, assunto)
```

```
3FN: Livros(nrol, título, editora, ano_public)
     Editoras(editora, cid_edit)
     AutAssLiv(nrol, autor, assunto)
```

Na 3FN, a base de dados ainda apresenta os seguintes problemas:

- Redundância para representar todas as informações;
- Representação não-uniforme (repete alguns elementos ou posições nulas).

Passagem à 4FN

- Geração de novas tabelas, eliminando dependências multivaloradas;
- Análise de dependências multivaloradas entre atributos:
  - *autor, assunto* → Dependência multivalorada de *nrol*.

Resultado

```
Livros(nrol, título, editora, ano_public)
Editoras(editora, cid_edit)
AutLiv(nrol, autor)
AssLiv(nrol, assunto)
```

Quinta Forma Normal

Está ligada à noção de dependência de junção.

- Se uma relação é decomposta em várias relações e a reconstrução não é possível pela junção das outras relações, dizemos que existe uma dependência de junção.
- Existem tabelas na 4FN que não podem ser divididas em duas relações sem que se altere os dados originais.

Exemplo: Sejam as relações R1(CodEmp, CodPrj) e R2(CodEmp, Papel) a decomposição da relação ProjetoRecurso(CodEmp, CodPrj, Papel).

Exemplo

- Da 4FN para a 5NF
- Explanação de que a última forma normal pode ser alcançada com projeções

Forma Normal De Boyce-Codd

Definição

> **Uma tabela está na BCNF se e somente se todo atributo não chave depender funcionalmente diretamente da chave primária, ou seja, não há dependências entre atributos não chave.** Porem nem toda tabela que está na 3FN é uma tabela BCNF.

Exemplo

- Como transformar da 3NF para BCNF
- Nem sempre pode ser alcançada preservando a dependência.

Sexta Forma Normal ou Forma Normal Chave-Domínio

Definição

De acordo com [1]Forma Normal Chave-Domínio é uma forma atualizada na normalização de banco de dados que necessita que não haja restrições de domínio e chave dentro do banco de dados. Esta forma[2] evita as restrições de dados que não são do domínio ou chaves com restrição. Muitos dos Banco de dados podem fazer os teste com seus domínios e chaves restritas com os seus atributos. Contudo a restrições necessitam da programação nesses bancos de dados.

Outras dependências

- dependências encapsuladas;
- dependências como blocos em lógica de primeira ordem.

Desnormalização

A criação de novas entidades e relacionamentos podem trazer prejuízos ao serem implementados em um **SGBD**. Devido a algumas relações excessivamente normalizadas, simples alterações no bancos de dados podem ocasionar uma profunda mudança na coerência de dados, assim entidades e relacionamentos devem ser desnormalizados para que o **SGBD** apresente um melhor desempenho.

Além disso, entidades podem conter ocorrências de mudanças de informações ao longo do tempo e a desnormalização pode contribuir com a manutenção de dados sem afetá-los drasticamente.

Para ilustrar o conceito podemos tomar como exemplo uma entidade denominada *Vendedor* com os seguintes atributos:

- Código do Vendedor;
- Nome do Vendedor;
- Região de Vendas.

Um vendedor cadastrado pode mudar sua área de vendas mas os dados de vendas antigas, realizadas em regiões por onde trabalhou, devem ser mantidos no banco sem incoerência de dados. Assim, a entidade *Vendedor* deve ser mantida desnormalizada para que os dados não se tornem inconsistentes.

Entidade Vendedor

Chave primaria	Codigo Vendedor	Nome	Regiao
001	132	Maria Auxiliadora	Natal
002	132	Maria Auxiliadora	Nova York

Nesse caso fomos obrigados a utilizar uma chave primária para manter a integridade de dados de *Vendedor*, o que mantém a entidade desnormalizada. Mas, ao optar pela desnomalização de dados devemos levar em conta o custo da redundância de dados e o uso das anomalias de atualização, o que não é interessante para grandes bancos de dados.

Ver também

- Arquitetura de dados
- Administração de dados
- Modelagem de dados
- Banco de Dados

Referências

1. Rodrigues, Lucas (27 de maio de 2014). «Domínio / forma normal chave (DKNF)» (https://prezi.com/7pq_quapedve/dominio-forma-normal-chave-dknf/). *prezi.com*. Consultado em 10 de dezembro de 2017
2. Rodrigues, Lucas (27 de maio de 2014). «Domínio / forma normal chave (DKNF)» (https://prezi.com/7pq_quapedve/dominio-forma-normal-chave-dknf/). *prezi.com*. Consultado em 10 de dezembro de 2017

Erro de citação: Elemento `<ref>` definido em `<references>` não tem um atributo de nome.
Erro de citação: Elemento `<ref>` definido em `<references>` não tem um atributo de nome.

Erro de citação: Elemento `<ref>` definido em `<references>` não tem um atributo de nome.

This article was originally based on material from the Free On-line Dictionary of Computing, used with permission. Update as needed.

Obtida de "https://pt.wikipedia.org/w/index.php?title=Normalização_de_dados&oldid=53943127"

Esta página foi editada pela última vez às 04h56min de 1 de janeiro de 2019.

Este texto é disponibilizado nos termos da licença Atribuição-CompartilhaIgual 3.0 Não Adaptada (CC BY-SA 3.0) da Creative Commons; pode estar sujeito a condições adicionais. Para mais detalhes, consulte as condições de utilização.