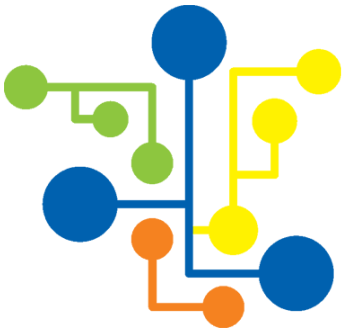

Data Science Academy

www.datascienceacademy.com.br

Design e Implementação de Data Warehouses

Mas o que é Normalização?

O Que é Normalização?

A normalização do banco de dados é o processo de transformações na estrutura de um banco de dados que visa a eliminar redundâncias e a eliminar anomalias de inserção, atualização e exclusão.

Ao efetuar o processo de normalização, os dados cadastrados no banco de dados ficarão organizados de uma forma melhor e ocuparão menos espaço físico. Entretanto, o processo de normalização também faz aumentar o número de tabelas e em muitos casos pode ser uma tarefa difícil de ser realizada. Além disso, bancos de dados normalizados além do necessário podem ter desempenho ruim e/ou complexidade excessiva.

A principal finalidade do processo de normalização é eliminar as anomalias de inserção, atualização e exclusão. A anomalia ocorre quando não há forma de se cadastrar alguma determinada informação sem que alguma outra informação também seja diretamente cadastrada. Por exemplo, imagine que você tenha uma tabela funcionário com os seguintes dados: codigo, nome, projeto; onde a tabela projeto corresponde ao nome do projeto no qual um funcionário foi alocado.

codigo	nome	projeto
1	Pedro	Vendas
2	Maria	Vendas
3	Carlos	Cadastro de clientes

E então surgiu um projeto novo, o de emissão de notas fiscais. Como você cadastra esse novo projeto? A resposta é que não dá para cadastrar, pois para fazer isso você teria que ter algum funcionário nesse projeto - ou seja, temos uma anomalia de inserção.

Se no exemplo anterior, o funcionário Carlos fosse desligado da empresa e o removermos da tabela, a informação sobre o projeto de cadastro de clientes é perdida. Isso é um efeito colateral indesejado - é a anomalia de exclusão. Se, no entanto, ele apenas fosse remanejado para o novo projeto de notas fiscais, nós também perderíamos a informação sobre a existência do projeto de cadastro de clientes - essa é a anomalia de alteração.

O problema que origina essas anomalias é o fato de a informação do projeto estar toda dentro da tabela de funcionários, que não é o lugar dela. Se tivermos duas tabelas relacionadas (1-para-N) - funcionários e projetos - as anomalias desaparecem. Anomalias também têm relação com o conceito de integridade referencial.

Integridade Referencial

Integridade referencial é um conceito relacionado a chaves estrangeiras. Este conceito diz que o valor que é chave estrangeira em uma tabela destino, deve ser chave primária de algum registro na tabela origem. Quando essa regra é desrespeitada, então temos o caso em que a integridade referencial é violada.

Vejamos a terminologia: Integridade vem de íntegro, inteiro, completo, correto. Referencial vem de referência, indicar algo ou alguém. Portanto, integridade referencial é indicar algo ou alguém de forma íntegra, completa, correta. Por exemplo, veja essas duas tabelas:

```
Carros
Placa (PK) | Modelo | Proprietário (FK)
-----------+--------+------------------
ABC-1233   | Passat | 1
DEF-4566   | Fiesta | 2
UUV-7890   | Palio  | 1

Proprietários
ID (PK) | Nome
--------+------
1       | Pedro
2       | Maria
```

Estas tabelas têm integridade referencial, pois os carros que têm proprietário com ID 1, podem ser encontrados na tabela de proprietários como sendo do Pedro. O carro de proprietário com ID 2 pode ser encontrado como sendo da Maria.

Agora, imagine que nós venhamos inserir um carro de placa EJB-6520, do modelo Civic e do proprietário com o ID 3. Ocorre que não há nenhum proprietário de ID 3. Se o banco de dados permitir essa inclusão, ocorrerá uma violação da integridade referencial, pois estará sendo feita uma referência a uma entidade inexistente. O mesmo ocorreria se quisermos alterar o proprietário de um dos carros colocando o ID do proprietário como 3.

Por outro lado, se nós quisermos deletar a Maria do banco de dados sem deletar o carro de placa DEF-4566 e nem alterá-lo, novamente teremos uma violação da integridade referencial, pois se o banco de dados permitir que essa exclusão seja feita, a integridade referencial será violada ao termos um carro que tem como dono, uma entidade agora inexistente.

A maioria dos bancos de dados relacionais modernos existentes impõe integridade referencial quando você tenta inserir, alterar ou excluir entidades no qual há chaves estrangeiras envolvidas. Se uma violação de integridade ocorrer, o seu banco de dados apresentará registros inconsistentes que apontam para entidades que não existem, o que tende a se manifestar nas aplicações sob a forma de vários tipos de problemas.

Formas Normais

Normalização de dados é um conjunto de regras aplicadas a tabelas de banco de dados relacionais a fim de manter a consistência dos dados, evitar duplicações/redundância e problemas com remoções ou atualizações de registros.

As formas normais são 1FN, 2FN, 3FN, BCNF, 4FN e 5FN. As tabelas geralmente são normalizadas até a terceira forma, a quarta e quinta formas normais tratam de problemas específicos. Vejamos aqui as 3 primeiras formas normais.

Antes de prosseguir com as formas normais, faz-se necessário introduzir os conceitos de chaves candidatas e superchaves.

A chave primária é aquele conjunto de colunas que serve para identificar a tupla de uma forma única (pode ser só uma coluna, ou podem ser duas ou mais). É importante que o projetista do banco de dados saiba identificar quais são as colunas mais apropriadas para serem eleitas como parte da chave primária.

Entretanto, às vezes há mais do que um conjunto de colunas que poderia ser chave primária. Cada um desses conjuntos é chamado de chave candidata. Por exemplo, em uma tabela Pessoa que tenha os campos CPF, RG, Estado e Nome, tanto o CPF quanto o RG junto com o Estado são chaves candidatas. Assim, é possível chegar-se ao Nome a partir do CPF, mas também é possível chegar-se ao Nome a partir do RG e do Estado.

Qualquer conjunto de colunas que tenha como subconjunto, uma chave candidata é denominado de superchave.

Primeira Forma Normal (1FN)

A primeira regra para eliminar as anomalias é: Não devem existir colunas multivaloradas. O que é uma coluna multivalorada? É uma coluna na qual é possível armazenar mais de um valor por registro. Por exemplo, imagine que você tenha na sua tabela de clientes as colunas código, nome e telefones, preenchidos assim:

```
codigo    nome      telefones
1         Paulo     99567-4289, 3605-4900
2         Maria     97223-0444
3         Alan      -
4         Juliana   2853-0266, 96610-5480, 2410-9941
```

Observe que a coluna de telefones é multivalorada. Inserir, atualizar e excluir telefones nesse esquema é algo complicado de se fazer. Pesquisar por algum número de telefone específico também é algo complicado. A solução para isso é dividir em duas tabelas:

```
Cliente
codigo    nome
1         Paulo
2         Maria
3         Alan
4         Juliana

Telefone
codigo    codigo_cliente   numero
1         1                99567-4289
2         1                3605-4900
3         2                97223-0444
4         4                2853-0266
5         4                96610-5480
6         4                2410-9941
```

Ao eliminar-se todas as colunas multivaloradas, o banco de dados atinge uma forma estrutural denominada de primeira forma normal, ou simplesmente 1FN.

Segunda Forma Normal (2FN)

A segunda forma normal é aquela que diz que: Todas as colunas devem ter dependência funcional com a totalidade de cada chave candidata.

Na maioria dos casos por "cada chave candidata", entenda-se por "com a chave primária", exceto se houver mais do que uma chave candidata.

Além disso, para que uma tabela esteja na segunda forma normal, ela deve estar antes de tudo, na primeira forma normal.

Uma coluna está em dependência funcional com a chave primária quando ela é determinada no domínio da aplicação por meio da chave primária. Uma coluna não tem dependência funcional com a chave primária quando ela é definida de forma independente da chave primária ou quando ela é definida a partir de alguma outra coluna que não seja a chave primária.

Uma dependência funcional pode ser dita estar na totalidade da chave primária quando todos os campos da chave primária são necessários para estabelecer a relação de dependência. No caso de a chave primária ser composta, é possível se ter uma dependência parcial.

Para dar um exemplo da 2FN, imagine que sua empresa tenha representantes comerciais atuando em clientes e queira representar a relação de quais representantes comerciais atuam em quais clientes. Nessa tabela (vamos chamar de representação), temos as colunas nome_cliente e nome_representante como chaves primárias e temos também as colunas endereço_representante, endereço_cliente e valor_contrato.

Observe que as colunas endereço_representante e endereço_cliente dependem da chave primária. Mas eles não dependem de toda a chave primária, cada um depende apenas de parte dela. A solução neste caso é ter uma tabela de clientes (com o endereço do cliente), uma tabela de representantes comerciais (com o endereço dele também) e deixar na tabela de atuação, as respectivas chaves estrangeiras com o valor do contrato.

Terceira Forma Normal (2FN)

A terceira forma normal é aquela que diz que: Todas as colunas devem ter dependência funcional com a totalidade de cada chave candidata e nada mais além do que essas chaves candidatas.

Novamente, na maioria dos casos por "cada chave candidata", entenda-se por "com a chave primária", exceto se houver mais do que uma chave candidata. Se a única chave candidata existente for a chave primária, isso ficaria assim: Todas as colunas devem ter dependência funcional com a totalidade da chave primária e nada mais além da chave primária.

Além disso, para uma tabela estar na terceira forma normal, ela deve estar primeiramente na segunda forma normal (e também na primeira). A parte de depender da totalidade de cada chave candidata é abordada na segunda forma normal, então o foco aqui é depender de nada mais que essas chaves.

Por exemplo. Imagine a tabela de carros com as colunas placa (chave primária), cor, nome_proprietário, endereço_proprietário:

placa	cor	nome_proprietário	endereço_proprietário
ABX-1234	azul	José	Rua X, 123
NNU-5566	verde	Marcos	Rua exemplo, 5678
SGH-7210	preto	Maria	Avenida teste, 7743
ERT-6902	vermelho	José	Rua X, 123
BGH-5431	preto	José	Rua X, 123

Observe o endereço do proprietário - ele é uma violação da terceira forma normal. Observe que o endereço do proprietário é definido como resultado de quem é o proprietário, e não como consequência da placa do carro. Se o José mudar de endereço e atualizarmos o endereço de apenas um dos carros dele, o banco de dados ficará inconsistente (há anomalia de alteração). Se a Maria comprar mais um novo carro e adicionarmos com um outro endereço, também ficará inconsistente (anomalia de inserção). Se o Marcos vender o carro dele, o seu endereço será esquecido (anomalia de exclusão). A solução novamente, é separar em duas tabelas:

Carro

placa	cor	proprietário
ABX-1234	azul	1
NNU-5566	verde	2
SGH-7210	preto	3
ERT-6902	vermelho	1
BGH-5431	preto	1

Proprietário

codigo	nome	endereço
1	José	Rua X, 123
2	Marcos	Rua exemplo, 5678
3	Maria	Avenida teste, 7743

Até onde normalizar?

Em geral, muitos ficam satisfeitos em atingir a terceira forma normal e não se preocupam muito com as demais formas normais superiores a essa porque se aplicam a casos raros, pois quando a terceira forma normal é atingida, quase sempre a Boyce-Codd, a quarta e a quinta também foram por sorte ou acidente. Isso ocorre porque é difícil se ter um caso na 3NF que não está na BCNF e também porque tabelas com três ou mais colunas na chave primária e que tenham relações de dependência entre essas colunas são algo bem raro.

Uma vez que a terceira forma normal é atingida (e provavelmente, por sorte a quarta e a quinta também), restam bem poucas possibilidades que permitam a introdução de anomalias. Procurar atingir a sexta forma normal é em geral exagero e não trás benefício prático algum. Procurar atingir a forma normal de domínio-chave é quase sempre impossível, embora essa busca possa revelar ainda alguma possibilidade de anomalia que possa ser eliminada.

Por vezes, para melhorar-se o desempenho do banco de dados ou simplificar-se a sua estrutura, a recomendação acaba sendo a de desnormalizar algumas coisas. Muitos sistemas com foco em Data Warehouse são normalizados só até a segunda forma normal, muitas vezes sendo concebidos a partir de estruturas que já estava na terceira forma normal e que sofreram uma desnormalização.

O conceito de normalização e as formas normais se aplicam somente aos bancos de dados relacionais. Outros tipos de bancos de dados que não sejam relacionais (por exemplo, uma aplicação que salve dados em arquivos organizados em pastas), também podem ter conceitos análogos a normalização para eliminar redundâncias, melhorar a estrutura e reduzir a possibilidade de produzirem anomalias. Entretanto, nesse campo, os conceitos de normalização não são tão bem definidos quanto no caso de bases de dados relacionais e cada caso terá as suas particularidades.

Vale ainda ressaltar que normalização de banco de dados consiste na normalização da estrutura do banco de dados, enquanto que normalização de dados corresponde à normalização dos dados já existentes nas tabelas.

Entretanto, especialmente em caso de reestruturações de bancos de dados que já estão em produção, os dois conceitos andam tão juntos e misturados que nem acaba fazendo sentido falar-se em um deles sem estar falando também do outro. Daí, em muitas situações que acontecem na prática, eles acabam sendo colocados como se fossem sinônimos.

Referências:

Data Warehouse Requirements Engineering: A Decision Based Approach
https://www.amazon.com.br/Data-Warehouse-Requirements-Engineering-Decision-ebook/dp/B079GT1JJ2/ref=tmm_kin_swatch_0?_encoding=UTF8&qid=1522639019&sr=8-4

Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design
https://www.amazon.com.br/Database-Design-Mere-Mortals-Hands-ebook/dp/B00BHEY5C2?keywords=database+modeling&qid=1522639134&sr=8-1&ref=sr_1_1

Conceptual Data Modeling and Database Design: A Fully Algorithmic Approach, Volume 1: The Shortest Advisable Path
https://www.amazon.com.br/Conceptual-Data-Modeling-Database-Design-ebook/dp/B015PNES0E?keywords=database+modeling&qid=1522639134&sr=8-3&ref=sr_1_3

Database Modeling and Design: Logical Design (The Morgan Kaufmann Series in Data Management Systems)
https://www.amazon.com.br/Database-Modeling-Design-Kaufmann-Management-ebook/dp/B004NNUZ6E?keywords=database+modeling&qid=1522639134&sr=8-16&ref=sr_1_16