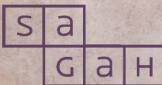


# BANCO DE DADOS

Roni Francisco Pichetti



SOLUÇÕES  
EDUCACIONAIS  
INTEGRADAS

# Engenharia reversa e normalização

## Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Descrever o processo de engenharia reversa em um banco de dados.
- Aplicar as regras de normalização em banco de dados.
- Reconhecer as anomalias em banco de dados e as limitações do modelo entidade-relacionamento (ER).

## Introdução

A construção dos modelos que formam a documentação de um banco de dados é importante para garantir o acompanhamento de suas mudanças ao longo do tempo, assim como para realizar manutenções e melhorias no sistema propriamente dito. Porém, há casos em que essa documentação não foi feita, ou não foi atualizada de acordo com as alterações realizadas no banco de dados, e, portanto, é necessário recriá-la. Para isso, utiliza-se a engenharia reversa de banco de dados, que pode ser tratada como uma metodologia ou um caminho a ser seguido para a construção de modelos de dados.

Neste capítulo, você vai estudar o processo de engenharia reversa em bancos de dados. Você também vai verificar exemplos sobre a aplicação das regras de normalização de banco de dados e compreender as possíveis anomalias e limitações do modelo entidade-relacionamento (ER).

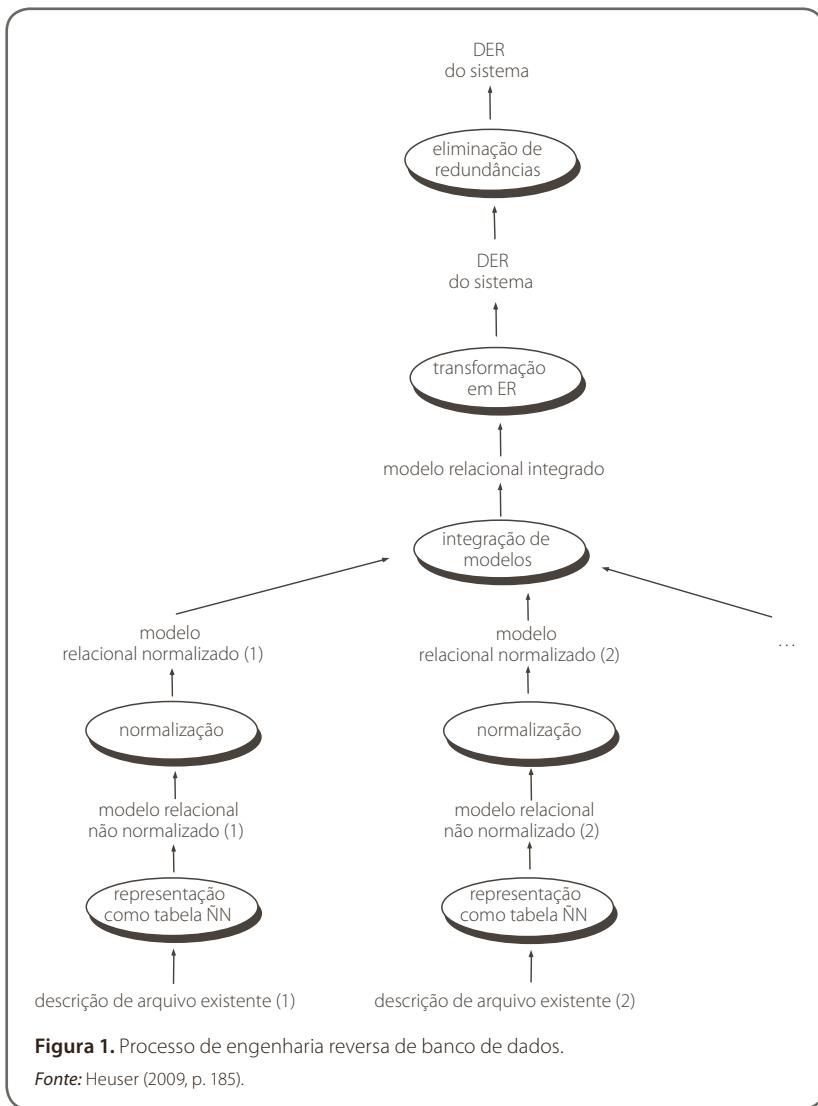
## 1 Engenharia reversa em um banco de dados

A engenharia reversa de banco de dados pode ser utilizada para a transformação entre modelos relacionais, como entre um modelo lógico e um conceitual. Serve também para os casos em que é necessário obter um modelo relacional de um modelo lógico de um banco de dados não relacional. Nesse caso, é realizada a engenharia reversa de arquivos, com base nos conjuntos de dados disponíveis sobre o banco de dados, sejam documentos, arquivos manuais ou um banco de dados gerenciado por um sistema de gerenciamento de banco de dados (SGBD) não relacional.

Por isso, a engenharia reversa pode ser utilizada em bancos de dados que não tenham a documentação em forma de um modelo conceitual, sejam eles relacionais ou não relacionais. Isso pode trazer benefícios em ambos os casos, visto a importância de se ter a documentação referente à modelagem de um banco de dados. Um exemplo disso é o caso de manutenções rotineiras no banco de dados. Um modelo conceitual pode ser utilizado para revisar com os usuários e desenvolvedores do banco de dados as suas funcionalidades, assim como para repassar esse conhecimento para pessoas que não conheçam o sistema, de forma rápida. Outro exemplo é a migração de um banco de dados para um SGBD relacional. Por meio da documentação abstrata relacional, esse processo pode ser mais rápido (HEUSER, 2009).

A engenharia reversa dos dados pode acontecer em diferentes níveis de abstração; em casos de reengenharia total de um sistema, costuma ser a primeira tarefa. Estruturas de dados como os bancos de dados podem necessitar passar por mudanças muitas vezes, como em uma ação de importação de arquivos de texto para SGBDs relacionais. Dessa forma, para realizar engenharia reversa em uma estrutura de banco de dados, é necessário compreender as relações entre os seus objetos (PRESSMAN; MAXIM, 2016).

Na Figura 1, é apresentada uma visão geral do processo de engenharia reversa de um banco de dados com arquivos convencionais. Nesse processo, pretende-se normalizar o banco de dados, o que quer dizer que será avaliada a possibilidade de aumentar o seu desempenho, reduzir a redundância de dados e aumentar a sua integridade. Mais detalhes sobre a normalização serão abordados na próxima seção.



**Figura 1.** Processo de engenharia reversa de banco de dados.

**Fonte:** Heuser (2009, p. 185).

O processo que está na Figura 1 inicia de baixo para cima, com a **descrição de cada um dos arquivos** que fazem parte do sistema existente. Nessa etapa, são obtidas as descrições independentes do tipo de arquivo que está sendo utilizado. Em seguida, o processo trabalha somente com **tabelas relacionais**, para que se torne independente do tipo de arquivo que está sendo utilizado como entrada no processo de normalização. Na sequência, o **modelo relacional** passa pelo processo de **normalização**, pelo qual se obtém um **modelo relacional normalizado**, com descrições das tabelas correspondentes ao arquivo sendo verificado (HEUSER, 2009).

Depois que todos os arquivos do sistema estiverem normalizados, os modelos obtidos são **integrados**, para gerar um esquema relacional do banco de dados como um todo. Com base no **esquema relacional** obtido no processo, é possível construir o **modelo ER** e o diagrama ER (DER) do sistema. Por fim, esse DER passa pelo processo de eliminação das redundâncias, que se trata de verificar as informações presentes em mais de um arquivo e representá-las uma só vez.

Para deixar claro como funciona o processo de engenharia reversa, veja um exemplo de documento com informações referentes à gerência de dois projetos. O documento não está normalizado, conforme se pode ver no Quadro 1.

**Quadro 1.** Documento a ser normalizado

Relatório de alocação a projeto					
Código do projeto: LSC001			Tipo: novo desenv.		
Descrição: sistema de estoque					
Código do colaborador	Nome	Categoria funcional	Salário	Data de início no projeto	Tempo alocado ao projeto
1100	Paulo	A1	5000	01/11/2018	24
1235	Roberto	A2	4000	02/10/2018	24
2123	Ana	B1	3500	03/10/2019	18
1435	Maria	A2	4000	04/10/2019	18
1854	Fernando	A1	5000	01/11/2019	12

(Continua)

(Continuação)

**Quadro 1.** Documento a ser normalizado**Relatório de alocação a projeto****Código do projeto:** PAG02**Tipo:** manutenção**Descrição:** sistema de recursos humanos (RH)

<b>Código do colaborador</b>	<b>Nome</b>	<b>Categoria funcional</b>	<b>Salário</b>	<b>Data de início no projeto</b>	<b>Tempo alocado ao projeto</b>
1854	Fernando	A1	5000	01/05/2020	12
1100	Paulo	A1	5000	04/01/2018	24
2123	Ana	B1	3500	01/11/2019	12

**Fonte:** Adaptado de Heuser (2009).

Em cada projeto existe um código, uma descrição, um tipo e dados sobre os seus colaboradores. Os dados dos colaboradores são: código, nome, categoria funcional, data de início de sua colaboração no projeto e tempo alocado para atuação no projeto.

O primeiro passo para aplicação da engenharia reversa no documento do Quadro 1 é a transformação de sua descrição em um esquema de uma tabela relacional, para que possa ser normalizado. No Quadro 2, a seguir, é apresentada essa tabela, que possui outra tabela aninhada; trata-se de uma tabela que está dentro de outra tabela como valor. No caso do Quadro 2, na coluna “Emp”, está presente uma tabela aninhada para cada departamento, com os dados dos colaboradores de cada setor envolvidos nos projetos relacionados.

**Quadro 2.** Tabela a ser normalizada

CodProj	Tipo	Descr	Empresa					
			CodEmp	Nome	Cat	Sal	DataIni	TempAl
LSC001	Novo desenv.	Sistema de estoque	1100	Paulo	A1	5000	01/11/2018	24
			1235	Roberto	A2	4000	02/10/2018	24
			2123	Ana	B1	3500	03/10/2019	18
			1435	Maria	A2	4000	04/10/2019	18
			1854	Fernando	A1	5000	01/11/2019	12
PAG02	Manutenção	Sistema de RH	1854	Fernando	A1	5000	01/05/2020	12
			1100	Paulo	A1	5000	04/01/2018	24
			2123	Ana	B1	3500	01/11/2019	12

Fonte: Adaptado de Heuser (2009).

Esse mesmo quadro pode ser representado pelo seguinte esquema relacional:

```
Proj (CodProj, Tipo, Descr,
      (CodEmp, Nome, Cat, Sal, DataIni, TempAl))
```

Nesse esquema, são utilizados parênteses aninhados, para deixar claro que existem tabelas aninhadas. Nas tabelas aninhadas, as colunas sublinhadas servem para diferenciá-las de linhas com informações parecidas que estão nas tabelas externas. No caso do exemplo, a coluna “CodProj” é distinta das linhas para cada projeto da tabela “Proj”, enquanto a coluna “CodEmp” distingue as linhas sobre os empregados no grupo de cada projeto.

Portanto, a engenharia reversa é vantajosa quando não há modelos de banco de dados ou modelo lógico existentes, ou quando um banco de dados necessita passar por modificações periódicas, mas elas não são documentadas. A engenharia reversa contribui para facilitar a compreensão do relacionamento entre as entidades presentes em um banco de dados, tanto por usuários quanto por administradores e desenvolvedores (HEUSER, 2009).

## 2 Normalização em banco de dados

Com o esquema relacional do documento, a próxima etapa da engenharia reversa de um banco de dados é a normalização. Essa etapa é baseada no conceito de **forma normal**, que consiste em uma regra que precisa ser obedecida em uma tabela até ela ser considerada “bem projetada”. Existe um número variado de formas normais que podem ser aplicadas. Neste capítulo, você vai estudar as primeiras quatro delas, que são abreviadas como 1FN, 2FN, 3FN e 4FN (HEUSER, 2009).

Para decidir se um esquema de relações de banco de dados é bom ou se necessita ser decomposto em relações menores, procuram-se possíveis problemas em sua utilização. Por isso, são utilizadas as formas normais, a fim de confirmar se determinados tipos de problemas podem surgir ou não no esquema proposto (RAMARKRISHNAN; GEHRKE, 2011).

A **primeira forma normal** (1FN) se preocupa em transformar o esquema de tabela não normalizado em um esquema relacional sem tabelas aninhadas. Então, para passar pela 1FN, as tabelas aninhadas, se houver, devem ser eliminadas. Dessa forma, a 1FN reprova a existência de um conjunto de valores, uma linha com valores ou a combinação de ambos, como o valor de atributo de determinada linha ou relação.

Isso quer dizer que a 1FN reprova relações dentro de relações, relações aninhadas ou, ainda, relações que sejam valores de atributo dentro de linhas. Ela permite somente **valores indivisíveis** nos atributos de uma tabela (ELMASRI; NAVATHE, 2011). De acordo com Heuser (2009), esse processo pode ser feito de duas formas, descritas a seguir.

1. Construção de uma tabela única com redundância de dados, na qual os dados de linhas externas da tabela aninhada são repetidos em cada linha da tabela aninhada. Levando-se em conta a tabela representada no Quadro 2, o resultado desse esquema resulta na repetição dos dados sobre o projeto em cada linha da tabela de empregados:

```
ProjEmp (CodProj, Tipo, Descr, CodEmp, Nome, Cat, Sal, DataIni,  
TempA1)
```

2. Construção de uma tabela para cada tabela aninhada. Assim, a tabela que está sendo normalizada é mantida, e são criadas mais tabelas para cada uma que esteja aninhada. Novamente se considerando a tabela representada no Quadro 2, o resultado seria duas tabelas diferentes:

```
Proj (CodProj, Tipo, Descr)  
ProjEmp (CodProj, CodEmp, Nome, Cat, Sal, DataIni, TempAl)
```

A opção de criar uma tabela única é a mais correta, quando levada em conta apenas a correção do processo de normalização, visto que a construção de diferentes tabelas pode ocasionar perda de relações entre informações. Porém, na prática, é preferível a utilização da segunda alternativa, mesmo que gere imperfeições nos modelos. Isso porque manter uma única tabela com muitas tabelas aninhadas dificulta a visualização da tabela na 1FN.

Segundo Heuser (2009, p. 192):

A passagem à primeira forma normal por decomposição de tabelas é feita nos seguintes passos:

- 1 É criada uma tabela 1FN referente à tabela não normalizada e que contém apenas as colunas com valores atômicos, isto é, sem as tabelas aninhadas. A chave primária da tabela na 1FN é idêntica à chave da tabela não normalizada.
- 2 Para cada tabela aninhada, é criada uma tabela na 1FN composta pelas seguintes colunas: a chave primária de cada uma das tabelas nas quais a tabela em questão está aninhada. As colunas da própria tabela aninhada.
- 3 São definidas as chaves primárias das tabelas na 1FN que correspondem a tabelas aninhadas.

Dessa forma, o esquema relacional com a criação de duas tabelas do exemplo anterior é:

## 1FN

```
Proj (CodProj, Tipo, Descr)  
ProjEmp (CodProj, CodEmp, Nome, Cat, Sal, DataIni, TempAl)
```

Em que a chave primária da tabela “Proj” é a coluna “CodProj”, enquanto, na tabela “ProjEmp”, utilizam-se duas colunas para a chave primária, “CodProj” e “CodEmp”, pois o mesmo empregado pode trabalhar em mais de um projeto. Por isso, na tabela “ProjEmp”, é utilizada mais de uma linha para o mesmo empregado, com o código do projeto para diferenciar.

As duas tabelas com seus respectivos dados são apresentadas no Quadro 3.

**Quadro 3.** Tabela de exemplo após a aplicação da 1FN

Proj						
CodProj	Tipo	Descr				
LSC001	Novo desenv.	Sistema				
PAG02	Manutenção	Sistema de RH				
ProjEmp						
Cod-Proj	CodEmp	Nome	Cat	Sal	DataIni	TempAl
LSC001	1100	Paulo	A1	5000	01/11/2018	24
LSC001	1235	Roberto	A2	4000	02/10/2018	24
LSC001	2123	Ana	B1	3500	03/10/2019	18
LSC001	1435	Maria	A2	4000	04/10/2019	18
LSC001	1854	Fernando	A1	5000	01/11/2019	12
PAG02	1854	Fernando	A1	5000	01/05/2020	12
PAG02	1100	Paulo	A1	5000	04/01/2018	24
PAG02	2123	Ana	B1	3500	01/11/2019	12

**Fonte:** Adaptado de Heuser (2009).

As próximas duas formas normais são relacionadas ao conceito de **dependência funcional**. Segundo Date (2003, p. 289), “[...] dependência funcional (em geral abreviada como DF) é um relacionamento de muitos para um entre um conjunto de atributos e outro, com respeito a determinada RelVar”. A abreviação “RelVar”, nesse caso, se refere à variável relacional. Isso quer dizer que as dependências funcionais representam restrições de integridade que o SGBD precisa impor, para que sejam aplicadas automaticamente (DATE, 2003).

Para compreender melhor o conceito de dependência funcional, considere o seguinte exemplo: a tabela “ProjEmp”, presente no Quadro 3, possui uma coluna para o salário e outra para o código do empregado. Pode-se dizer que a coluna “Sal” depende funcionalmente da coluna “CodEmp”, ou ainda que a coluna do código do empregado determina a coluna do seu salário — isso porque o valor de “CodEmp” está sempre vinculado ao mesmo valor de “Sal”. Essa dependência é representada da seguinte forma: “CodEmp” →

“Sal”, onde a coluna “CodEmp” é o determinante da dependência funcional. Na prática, um determinante de uma dependência funcional pode ser uma coluna ou um conjunto de colunas.

Nesse sentido, a passagem pela **segunda forma normal** (2FN) elimina redundâncias de dados. Por exemplo, pense novamente na tabela “ProjEmp”. Nela, estão os dados sobre os empregados de cada projeto. Porém, quando o empregado se repete, dados como nome, categoria e salário são repetidos — portanto, são redundantes, o que é eliminado por meio do uso da 2FN.

Para estar na 2FN, uma tabela precisa estar na 1FN, assim como as colunas que não são chaves devem depender da chave primária completa. Ou seja, uma tabela na qual as colunas que não são chave dependem somente de uma parte da chave primária não passou pela 2FN. Esse tipo de dependência é chamado de **dependência funcional parcial**. Cabe deixar claro que isso somente é possível em casos em que a tabela possui mais de uma coluna como chave primária. Quando há somente uma chave primária, a tabela não poderá conter dependências parciais, apenas completas (ELMASRI; NAVATHE, 2011).

Uma dependência parcial pode ser observada na tabela “ProjEmp”, pois sua chave primária é composta por duas colunas. Isso porque as colunas “Nome”, “Cat” e “Sal” dependem exclusivamente da coluna “CodEmp”, pois são informações que podem ser relacionadas somente ao código do empregado. Entretanto, as colunas “DataIni” e “TempA1” dependem da chave primária como um todo, pois essas informações se relacionam tanto com o código do empregado quanto com o código do projeto (HEUSER, 2009).

Para eliminar as dependências parciais, nesse caso, será necessário dividir a tabela “ProjEmp” em duas tabelas. Uma permanece com a descrição original, e a segunda passa a se chamar somente “Emp”, conforme o esquema relacional a seguir:

## 2FN

```
Proj (CodProj, Tipo, Descr)
ProjEmp (CodProj, CodEmp, DataIni, TempA1)
Emp (CodEmp, Nome, Cat, Sal)
```

O resultado das tabelas do exemplo após a passagem pela 2FN é apresentado no Quadro 4, que conta com a criação da nova tabela “Emp” e a eliminação das redundâncias dos dados sobre os empregados. A tabela “Proj” permanece igual.

**Quadro 4.** Tabela de exemplo após a aplicação da 2FN

<b>ProjEmp</b>			
CodProj	CodEmp	DataIni	TempAl
LSC001	1100	01/11/2018	24
LSC001	1235	02/10/2018	24
LSC001	2123	03/10/2019	18
LSC001	1435	04/10/2019	18
LSC001	1854	01/11/2019	12
PAG02	1854	01/05/2020	12
PAG02	1100	04/01/2018	24
PAG02	2123	01/11/2019	12
<b>Emp</b>			
CodEmp	Nome	Cat	Sal
1100	Paulo	A1	5000
1235	Roberto	A2	4000
2123	Ana	B1	3500
1435	Maria	A2	4000
1854	Fernando	A1	5000

*Fonte:* Adaptado de Heuser (2009).

O próximo passo é a passagem pela **terceira forma normal** (3FN), para eliminação de mais redundâncias. No exemplo que você está acompanhando, ainda é possível verificar redundância na tabela “Emp”, que está no Quadro 4. Visto que a informação do salário é relacionada à categoria funcional do empregado (coluna “Cat”), somente a informação da categoria já é suficiente para saber qual é o seu salário, sem a necessidade de uma coluna específica para esse fim.

Além da eliminação de redundâncias, a 3FN é utilizada no sentido de eliminar dependências funcionais transitivas ou indiretas. A **dependência funcional transitiva** ocorre quando uma coluna que não é chave primária

depende funcionalmente de uma ou mais colunas que também não são parte da chave. Portanto, a 3FN admite que as colunas tenham uma relação de dependência somente com a chave primária de sua tabela (HEUSER, 2009).

Para passar pela 3FN, é necessário dividir as tabelas com dependências funcionais transitivas, para eliminá-las. Assim, continuando o exemplo anterior, a 3FN gera o seguinte esquema relacional:

### 3FN

```
Proj (CodProj, Tipo, Descr)
ProjEmp (CodProj, CodEmp, DataIni, TempA1)
Emp (CodEmp, Nome, Cat)
Cat (Cat, Sal)
```

As tabelas referentes à passagem da 3FN são apresentadas no Quadro 5, com o desmembramento dos valores de salário da tabela “Emp” e a criação da tabela “Cat”. Isso resulta em diminuição da redundância das informações sobre os salários dos empregados, assim como elimina a dependência funcional transitiva da tabela “Emp”. As tabelas “Proj” e “ProjEmp” permanecem iguais à forma normal anterior.

**Quadro 5.** Tabela de exemplo após a aplicação da 3FN

Emp		
CodEmp	Nome	Cat
1100	Paulo	A1
1235	Roberto	A2
2123	Ana	B1
1435	Maria	A2
1854	Fernando	A1

Cat	
Cat	Sal
A1	5000
A2	4000
B1	3500

*Fonte:* Adaptado de Heuser (2009).

Geralmente, a decomposição até a 3FN é suficiente para a obtenção de um esquema de banco de dados que coincida com o arquivo ou documento utilizado. Porém, na engenharia reversa de banco de dados, é possível colocar em prática uma **quarta forma normal** (4FN). Ela é utilizada nos casos em que uma tabela já tenha sido validada, respectivamente:

- na 1FN e não tenha tabelas aninhadas;
- na 2FN e não possua dependência funcional parcial; e
- na 3FN e não apresente dependência funcional transitiva.

A 4FN é baseada na verificação de dependência funcional multivalorada. Isso acontece em casos em que uma coluna ou conjunto de colunas depende multivaloradamente de uma coluna, chamada de **determinante**, na mesma tabela da qual faz parte. Assim, um valor de atributo da coluna determinante é identificado de forma repetida em um conjunto de valores na coluna dependente (ELMASRI; NAVATHE, 2011).

Como exemplo de aplicação da 4FN, pense em uma tabela que relate o código do projeto (CodProj), o código do empregado (CodEmp) que faz parte do projeto e, ainda, o código de um equipamento utilizado por esse empregado no projeto (CodEquip). Essa tabela é chamada de “Utilização”. Nesse exemplo, as três tabelas fazem parte da chave primária. Na Figura 2 é apresentada a dependência multivalorada presente na tabela “Utilização”.

CodProj	CodEmp	CodEquip
1	1	1
1	2	1
1	3	1
1	1	2
1	2	2
1	3	2
2	2	-

**Figura 2.** Dependência multivalorada.

Fonte: Heuser (2009, p. 206).

Como se pode ver na Figura 2, a coluna “CodEmp” depende multivaloradamente da coluna “CodProj”, pois o valor 1 em “CodProj” determina múltiplas vezes um conjunto de valores da coluna “CodEmp”. Para que a tabela “Utilização” passe pela 4FN, ela precisa ser decomposta em duas tabelas (HEUSER, 2009). O esquema relacional fica assim:

```
ProjEmp (CodProj, CodEmp)  
ProjEquip (CodProj, CodEquip)
```

Dessa forma, a tabela deixa de ter informações redundantes sendo armazenadas, e a dependência multivalorada é eliminada, tornando a tabela “Utilização” normalizada pela 4FN.

### 3 Anomalias em banco de dados e limitações do modelo entidade-relacionamento

Agora que você conheceu quatro formas normais que são utilizadas na engenharia reversa de banco de dados, precisa saber que o processo de normalização também pode passar por problemas ou limitações. Entre eles estão a omissão de chaves primárias, a representação implícita de atributos que são relevantes para as tabelas ou, ainda, a existência de atributos irrelevantes, derivados ou redundantes.

A utilização de **chave primária** não é obrigatória em arquivos convencionais, da forma como se utiliza na abordagem relacional. Por isso, é possível que arquivos ou documentos que estejam disponíveis para normalização não apresentem uma chave primária. Por exemplo, em um documento com dados sobre colaboradores de uma organização, enviado para fiscalização para um órgão governamental, pode estar omitido o código do colaborador utilizado internamente, mas que não precisa ser fiscalizado. Se esse documento fosse utilizado para fins de normalização, não apresentaria a chave primária usual do banco de dados, o que tornaria necessário utilizar outra, como o CPF do colaborador. De qualquer forma, essa solução teria como resultado um modelo diferente do utilizado nas entidades representadas (HEUSER, 2009).

Quanto aos atributos, eles podem estar representados implicitamente em arquivos convencionais pela forma de ordenação de registros ou de listas, por exemplo. Nesses casos, deve-se agir como se o atributo aparecesse de forma explícita no arquivo. Um exemplo desse caso é uma tabela com o registro de alunos de uma universidade, os quais foram incluídos nessa tabela de acordo com a sua classificação no vestibular. Assim, o seu código de matrícula, que é gerado por autoincremento, foi criado nessa ordem. Porém, caso essa tabela passe por um processo de normalização, essa informação implícita da ordem das informações pode ser perdida. Por isso, o procedimento correto é inserir de forma explícita todas as informações necessárias sobre os atributos e as colunas da tabela no seu esquema relacional.

Outro problema que pode ocorrer é a presença de **atributos irrelevantes**, do ponto de vista conceitual, em arquivos convencionais. Eles podem existir por questões técnicas ou de desempenho do banco de dados em sua implementação. Alguns exemplos desses atributos são campos com o número de ocorrências de listas ou com o tamanho de outros campos. Nesses casos, os campos precisam ser eliminados desde sua passagem à forma não normalizada, antes de passar pela 1FN (HEUSER, 2009).

Algumas anomalias em bancos de dados podem ser causadas pela **redundância de dados**. Entre elas, estão (RAMARKRISHNAN; GEHRKE, 2011):

- anomalias de atualização, que ocorrem quando nem todos os dados redundantes são atualizados, o que gera inconsistências;
- anomalias de inserção, vinculadas a problemas na inclusão de informações em partes que necessitariam ser inseridas ao mesmo tempo em campos relacionados; e
- anomalias de exclusão, em casos em que, ao se excluir uma informação, ela acaba ocasionando a perda de demais informações relacionadas.

Levando-se em conta os problemas e as anomalias em bancos de dados causados pela possível fragilidade de seus modelos, cabe ressaltar a importância da aplicação das **normas funcionais**. Elas contribuem para a redução de redundâncias de informações no banco de dados, que, além de afetarem o desempenho do sistema, podem causar algumas das dificuldades listadas anteriormente. Nesse sentido, a normalização de cada documento ou arquivo de um sistema produz um conjunto de tabelas.

Após esse processo, outra etapa da engenharia reversa é a **integração dos modelos**, formando um **modelo global do banco de dados**. Seus principais objetivos são:

- juntar em uma única tabela os relacionamentos, as entidades e as colunas iguais que fizerem parte de diferentes tabelas;
- eliminar a redundância não somente de cada tabela de forma individual, mas das tabelas como um todo durante o processo de integração.

O processo de integração de modelos pode ser feito em três passos diferentes: integração de tabelas com chaves primárias iguais, integração de tabelas com chave contida e verificação de 3FN. Quanto à junção de tabelas com a **mesma chave primária**, exige-se que tanto os domínios quanto os conteúdos das respectivas colunas sejam iguais. Somente com essas duas coincidências pode acontecer esse tipo de junção (HEUSER, 2009). Veja um exemplo desse tipo de integração a seguir.



## Exemplo

Documento 1 já normalizado:

```
Proj (CodProj, Tipo, Descr)
ProjEmp (CodProj, CodEmp, DataIni, TempAl)
Emp (CodEmp, Nome, Cat)
Cat (Cat, Sal)
```

Documento 2 já normalizado:

```
Proj (CodProj, DataInicio, Descr, CodDepto)
Depto (CodDepto, NomeDepto)
ProjEquipamento (CodProj, CodEquipam, DataIni)
ProjEmp (CodProj, CodEmp, FunçãoEmpProj)
Equipamento (CodEquipam, Descrição)
```

Modelo global, integrado:

```
Proj (CodProj, Tipo, Descr, DataInicio, CodDepto)
ProjEmp (CodProj, CodEmp, DataIni, TempAl, FunçãoEmpProj)
Emp (CodEmp, Nome, Cat)
Cat(Cat, Sal)
Depto (CodDepto, NomeDepto)
ProjEquipamento (CodProj, CodEquipam, DataIni)
Equipamento (CodEquipam, Descrição)
```

**Fonte:** Heuser (2009, p. 210-211).

Perceba que a coluna “Tipo” teve origem no documento 1, as colunas “DataInicio” e “CodDepto” são do documento 2, e a coluna “Descr” estava nos dois documentos. Por fim, todas fazem parte do modelo integrado. Esse processo de integração de modelos é baseado na comparação entre os nomes de colunas e de tabelas; por isso, podem acontecer **homônimos**, quando dois objetos diferentes possuem o mesmo nome, e **sinônimos**, quando o mesmo objeto possui dois nomes diferentes. Esses conflitos são resolvidos com a renomeação das colunas.

A **integração de tabelas com chaves contidas** acontece quando uma tabela possui apenas a chave primária, a qual é subconjunto da chave primária de outra tabela, com mesmo domínio e valores (HEUSER, 2009). Como exemplo, considere uma tabela com a informação de que um aluno cursou uma disciplina e outra tabela com a informação da nota desse aluno em uma disciplina em um semestre. O esquema relacional das duas tabelas é este:

```
AlunoDisc (CodAl, CodDisc)
AlunoDiscSem (CodAl, CodDisc, SemDiscCursada, NotaDisc)
```

Considerando-se o exemplo, as colunas “CodAl” e “CodDisc” da primeira tabela possuem os mesmos valores da segunda tabela. Nesse caso, a tabela “AlunoDisc” é redundante, e a sua eliminação não causa perda de informações. Portanto, nesse caso, a integração seria utilizar somente a segunda tabela.

Porém, há casos em que isso não pode acontecer, pois, mesmo com o mesmo nome e conteúdo, duas colunas em tabelas diferentes podem ter importância para o conjunto de informações do banco de dados. Ou, ainda, uma tabela pode conter informações redundantes, mas que, se retiradas da tabela, fazem com que as demais colunas não tenham a funcionalidade esperada. Nesse contexto, é possível que a integração de duas tabelas que já passaram pela 4FN construa um modelo que está somente na 3FN. Por isso, depois da integração de modelos, é importante verificar as tabelas, para garantir que elas estejam cumprindo a 3FN de forma efetiva (HEUSER, 2009).

Seguindo as etapas da engenharia reversa de banco de dados apresentada na primeira seção deste capítulo, após a normalização e a integração de modelos, é necessário transformar o modelo integrado em modelos e diagramas ER. Resta deixar claro que a normalização não produz um modelo ER perfeito, visto que esse processo apenas elimina os campos multivvalorados e a redundância de dados observados com a aplicação das formas normais. Portanto, o último passo da engenharia reversa é verificar o modelo ER construído, corrigindo anomalias ou limitações ainda existentes.

Apesar dos métodos de engenharia reversa e normalização apresentados não tornarem perfeito o modelo e, posteriormente, o DER criado, é importante frisar que eles auxiliam na prevenção de diversos contratemplos que podem acontecer nesse processo. Por isso, quando for necessário construir modelos conceituais ou até mesmo lógicos, com base em documentos, arquivos ou mesmo um sistema, a utilização das formas normais aprimora em alto grau o resultado final.



### Saiba mais

Você viu ao longo deste capítulo diferentes conceitos sobre engenharia reversa de banco de dados e normalização. Pesquise por “Descrição das noções básicas de normalização de banco de dados” no seu mecanismo de busca e acesse a página da Microsoft que traz conceitos adicionais sobre a normalização de dados em ferramentas Microsoft (MICROSOFT, 2020).



## Referências

- DATE, C. J. *Introdução a sistemas de banco de dados*. 8. ed. Rio de Janeiro: Elsevier, 2003.
- ELMASRI, R.; NAVATHE, S. B. *Sistemas de banco de dados*. 6. ed. São Paulo: Pearson, 2011.
- HEUSER, C. A. *Projeto de banco de dados*. 6. ed. Porto Alegre: Bookman, 2009. (Série Livros didáticos informática UFRGS, v. 4).
- MICROSOFT. *Descrição das noções básicas de normalização de banco de dados*. [S. l.], 2020. Disponível em: <https://docs.microsoft.com/pt-br/office/troubleshoot/access/database-normalization-description>. Acesso em: 14 maio 2020.
- PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software: uma abordagem profissional*. 8. ed. Porto Alegre: AMGH, 2016.
- RAMAKRISHNAN, R.; GEHRKE, J. *Sistemas de gerenciamento de banco de dados*. 3. ed. Porto Alegre: AMGH, 2011.



## Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

