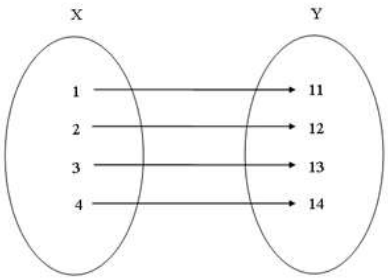
Normalização e Dependência Funcional

Dependência Funcional

O Modelo Relacional pegou emprestado da teoria de funções da matemática o conceito de dependência funcional.

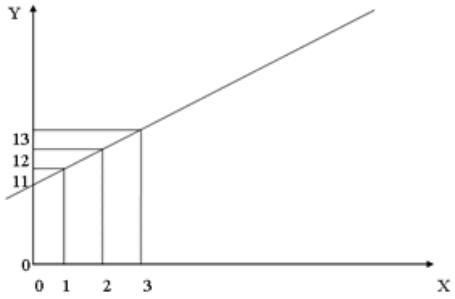
Iremos utilizar então a teoria de funções para explicar a dependência funcional do Modelo Relacional.

Considerando os seguintes conjuntos:

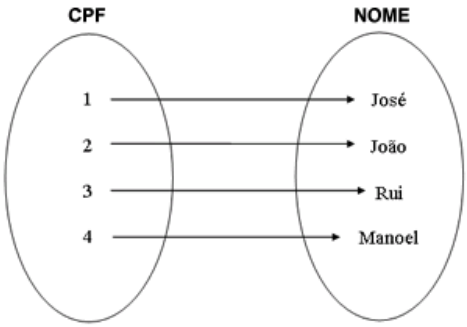


Observe que existe uma dependência entre os valores dos conjuntos, que pode ser expressa pela função f(x) = x + 10, ou seja, y é função de x, ou seja, y = f(x) = x + 10.

Esta dependência, esta função pode também ser expressa através do gráfico abaixo:



Agora, observe os conjuntos abaixo:



Observe que existe uma dependência entre os valores dos conjuntos, que pode ser expressa pela função f(CPF)=nome.

Ou seja, nome é função do CPF, ou seja, se eu tiver um número de CPF, poderei encontrar o nome da pessoa correspondente.

É claro que não existe uma figura gráfica que possa descrever esta função, mas ela existe.

Esta dependência é expressa no Modelo Relacional da seguinte maneira:

CPF -> NOME

Leia-se a notação acima das seguintes maneiras:

com um número de CPF eu posso encontrar o nome da pessoa, ou ainda:

nome depende funcionalmente do CPF.

Regras Para Encontrar Dependências Funcionais

1. Separação

A -> BC então A -> B e A -> C

Exemplo:

CPF -> nome, endereço então CPF -> nome e CPF -> endereço

Leia o exemplo acima da seguinte maneira:

Se com um número de CPF eu encontro o nome e o endereço de uma pessoa, então com este mesmo número eu posso encontrar apenas o nome e com este mesmo número eu posso encontrar apenas o endereço.

2. Acumulação

A -> B então AC -> B

Exemplo:

CPF -> endereço então CPF, idade -> endereço

Leia o exemplo acima da seguinte maneira:

Se com um número de CPF eu encontro o endereço de uma pessoa, então com este mesmo número mais a idade da pessoa eu posso encontrar o endereço também.

3. Transitividade

A -> B e B -> C então A -> C

Exemplo:

CPF -> código-cidade e código-cidade -> nome-cidade então CPF -> nome-cidade

Leia o exemplo acima da seguinte maneira:

Se com um número de CPF eu encontro o código da cidade de uma pessoa, e com o código da cidade eu encontro o nome da cidade, então com o número do CPF eu posso encontrar o nome da cidade.

4. Pseudo-Transitividade

A -> B e BC -> D então AC -> D

Exemplo:

CPF -> código-funcionário e código-funcionário, mês -> salário-funcionário então
CPF, mês -> salário-funcionário

Leia o exemplo acima da seguinte maneira:

Se com um número de CPF eu encontro o código do funcionário, e com o código do funcionário mais um certo mês eu encontro o salário que ele recebeu naquele mês, então com o número do CPF mais um certo mês eu posso encontrar o salário que ele recebeu naquele mês.

Formas Normais

O conceito de normalização foi introduzido por E. F. Codd em 1972.

Inicialmente Codd criou as três primeiras formas de normalização chamando-as de: primeira forma normal (1NF), segunda forma normal (2NF) e terceira forma normal (3NF). Uma definição mais forte da 3NF foi proposta depois por Boyce-Codd, e é conhecida como forma normal de Boyce-Codd (FNBC).

Através do processo de normalização pode-se, gradativamente, substituir um conjunto de entidades e relacionamentos por um outro, o qual se apresenta "purificado" em relação às anomalias de atualização (inclusão, alteração e exclusão) as quais podem causar certos problemas, tais como:

- Grupos repetitivos (atributos multivalorados) de dados;

- Variação temporal de certos atributos, dependências funcionais totais ou parciais em relação a uma chave concatenada;

- Redundâncias de dados desnecessárias;

- Perdas acidentais de informação;

- Dificuldade na representação de fatos da realidade observada;

- Dependências transitivas entre atributos.

Normalização de relações é portanto uma técnica que permite depurar um projeto de banco de dados, através da identificação de inconsistências (informações em duplicidade, dependências funcionais mal resolvidas, etc).

À medida que um conjunto de relações passa para uma forma normal, vamos construindo um banco de dados mais confiável.

O objetivo da normalização não é eliminar todos as inconsistências, e sim controlá-las.

Primeira Forma Normal

Uma relação está na primeira forma normal se todos os seus atributos são monovalorados e atômicos.

Quando encontrarmos um atributo multivalorado, deve-se criar um novo atributo que individualize a informação que esta multivalorada:

BOLETIM = {matricula-aluno, matéria, notas}

No caso acima, cada nota seria individualizada identificando a prova a qual aquela nota se refere:

BOLETIM = {matricula-aluno, matéria, número-prova, nota}

Quando encontrarmos um atributo não atômico, deve-se dividi-lo em outros atributos que sejam atômicos:

PESSOA = {CPF, nome-completo}

Vamos supor que, para a aplicação que utilizará esta relação, o atributo nome-completo não é atômico, a solução então será:

PESSOA = {CPF, nome, sobrenome}

Segunda Forma Normal

Uma relação está na segunda forma normal quando duas condições são satisfeitas:

1. A relação estiver na primeira forma normal;

2. Todos os atributos primos dependerem funcionalmente de toda a chave primária.

Observe a relação abaixo:

BOLETIM = {matricula-aluno, codigo-materia, numero-prova, nota, data-da-prova, nome-aluno, endereço-aluno, nome-materia}

Fazendo a análise da dependência funcional de cada atributo primo, chegamos às seguintes dependências funcionais:

- matricula-aluno, codigo-materia, numero-prova -> nota
- codigo-materia, numero-prova -> data-da-prova
- matricula-aluno -> nome-aluno, endereço-aluno
- codigo-materia -> nome-materia

Concluimos então que apenas o atributo primo nota depende totalmente de toda chave primária. Para que toda a relação seja passada para a segunda forma normal, deve-se criar novas relações, agrupando os atributos de acordo com suas dependências funcionais:

BOLETIM = {matricula-aluno, codigo-materia, numero-prova, nota}

PROVA = {codigo-materia, numero-prova, data-da-prova}

ALUNO = {matricula-aluno, nome-aluno, endereço-aluno}

MATERIA = {codigo-materia, nome-materia}

O nome das novas relações deve ser escolhido de acordo com a chave.

Terceira Forma Normal

Uma relação está na terceira forma normal quando duas condições forem satisfeitas:

1. A relação estiver na segunda forma normal;

2. Todos os atributos primos dependerem não transitivamente de toda a chave primária.

Observe a relação abaixo:

PEDIDO = {numero-pedido, codigo-cliente, data-pedido, nome-cliente,codigo-cidade-cliente, nome-cidade-cliente}

Fazendo a análise da dependência funcional de cada atributo primo, chegamos às seguintes dependências funcionais:

- numero-pedido -> codigo-cliente
- numero-pedido -> data-pedido
- codigo-cliente -> nome-cliente
- codigo-cliente -> codigo-cidade-cliente
- codigo-cidade-cliente -> nome-cidade-cliente

Concluimos então que apenas os atributos primos codigo-cliente e data-pedido dependem não transitivamente totalmente de toda chave primária.

Observe que:

- numero-pedido -> codigo-cliente -> nome-cliente
- numero-pedido -> codigo-cliente -> codigo-cidade-cliente
- numero-pedido -> codigo-cliente -> codigo-cidade-cliente -> nome-cidade-cliente

Isto é dependência transitiva, devemos resolver inicialmente as dependências mais simples, criando uma nova relação onde codigo-cliente é a chave, o codigo-cliente continuará na relação PEDIDO como atributo primo, porém, os atributos que dependem dele devem ser transferidos para a nova relação:

PEDIDO = {numero-pedido, codigo-cliente, data-pedido}

CLIENTE = {codigo-cliente, nome-cliente, codigo-cidade-cliente, nome-cidade-cliente}

As dependências transitivas da relação PEDIDO foram eliminadas, porém ainda devemos analisar a nova relação CLIENTE:

codigo-cliente -> codigo-cidade-cliente -> nome-cidade-cliente

Observe que o nome-cidade-cliente continua com uma dependência transitiva, vamos resolvê-la da mesma maneira :

PEDIDO = {numero-pedido, codigo-cliente, data-pedido}

CLIENTE = {codigo-cliente, nome-cliente, codigo-cidade-cliente}

CIDADE = {codigo-cidade-cliente, nome-cidade-cliente}

O nome das novas relações deve ser escolhido de acordo com a chave.