Cruzeiro do Sul
Educacional

Implementando o Modelo Relacional

Conteudista
Prof. Me. Hugo Fernandes

Revisão Textual
Prof.ª Alana Souza

Sumário

Sumário **2**

Objetivos da Unidade **3**

Normalização **4**

A Necessidade de Normalização 4

Anomalias 5

1ª Forma Normal (1FN) 7

2ª Forma Normal (2FN) 9

3ª Forma Normal (3FN) 11

Forma Normal Boyce/Codd (FNBC) e 4ª e 5ª Forma Normal (4FN, 5FN) 13

Em Síntese **15**

Atividades de Fixação **16**

Material Complementar **17**

Referências **18**

Gabarito **19**

Objetivos da Unidade

- Conceituar o Modelo Relacional;
- Definir Tabelas, atributos, domínio, tuplas, chave primária e estrangeira;
- Estudaremos os conceitos de Normalização do Modelo Relacional, suas regras de implantação e seus principais benefícios.

Atenção, estudante! Aqui, reforçamos o acesso ao conteúdo *on-line* para que você assista à videoaula. Será muito importante para o entendimento do conteúdo.

Este arquivo PDF contém o mesmo conteúdo visto *on-line*. Sua disponibilização é para consulta *off-line* e possibilidade de impressão. No entanto, recomendamos que acesse o conteúdo *on-line* para melhor aproveitamento.

VOCÊ SABE RESPONDER?

Para darmos início à unidade, contudo, convidamos você a refletir acerca da questão: você sabe como é feita a implementação do Modelo Relacional?

Normalização

A normalização é um processo para avaliar e corrigir estruturas de tabela para minimizar redundâncias de dados, reduzindo, assim, a probabilidade de anomalias de dados.

O processo de normalização envolve a incumbência de atributos a tabelas com base no conceito de determinação. O processo de normalização deve ocorrer logo após a etapa de criação do modelo conceitual do banco de dados. Muitas vezes, após a normalização, ocorrerão atualizações no modelo conceitual.

A normalização funciona por meio de uma série de estágios chamados Formas Normais. Os três primeiros estágios são descritos como primeira forma normal (1NF), segunda forma normal (2NF) e terceira forma normal (3NF). Do ponto de vista estrutural, podemos afirmar que 2NF é melhor do que 1NF, e 3NF é melhor que 2NF.

Importante

Um problema óbvio com informações redundantes é que usamos mais memória do que é necessário. A redundância é um exemplo de uma anomalia que pode ocorrer em um SGBD do modelo relacional.

A Necessidade de Normalização

Existem duas situações comuns em que os projetistas de banco de dados usam a normalização:

- Ao projetar uma nova estrutura de banco de dados com base nos requisitos de negócios dos usuários finais, o projetista de banco de dados construirá um

modelo de dados usando o diagrama de entidade-relacionamento (DER). Após a conclusão do projeto inicial, o projetista pode usar a normalização para analisar as relações que existem entre os atributos dentro de cada entidade, para determinar se a estrutura pode ser melhorada por meio da normalização;

- Alternativamente, os projetistas de banco de dados são frequentemente solicitados a modificar estruturas de dados existentes que podem ser na forma de arquivos simples, planilhas ou estruturas de banco de dados mais antigas. Novamente, por meio de uma análise das relações entre os atributos ou campos na estrutura de dados, o projetista de banco de dados pode utilizar o processo de normalização para melhorar a estrutura de dados existente para criar um projeto de banco de dados apropriado.

Anomalias

Existem três tipos de anomalias que ocorrem quando o banco de dados não é normalizado. Estas são:

- Inserção;
- Atualização;
- Anomalia de Exclusão.

Vamos a um exemplo para entender isso! Suponha que em uma empresa desenvolvedora de software, as informações dos projetos executados são armazenadas em uma tabela chamada Projetos. Vejamos essa tabela abaixo:

Tabela 1

ID_ projeto	Nome_ Projeto	ID_ Empregado	Nome Empregado	Cargo_ Empregado	Valor_ Hora	Horas_ Trabalhadas
001	Manhattan	1	João da Silva	Programador Sênor	40.00	50
001	Manhattan	2	Paulo Farias	Analista Sênior	40,00	30
001	Manhattan	3	Carlos Alberto	Programador Sênior	80,00	50’
001	Manhattan	4	Maria Fernanda	Gerente	80,00	50

Os problemas com a tabela acima são:

1. O número do projeto destina-se a ser uma chave primária, mas contém nulos;
2. A tabela exibe redundâncias de dados;
3. As entradas de tabela permitem inconsistências de dados;
4. As redundâncias de dados produzem as seguintes anomalias:

Anomalias de Inserção

Não podemos armazenar os detalhes do Empregado até que o Projeto seja atribuído.

Anomalias de Atualização

Se o nome do projeto Manhattan precisar ser alterado, essa alteração deverá ser realizada em todos os quatro registros.

Anomalias de Exclusão

Se excluirmos o empregado 1, também perderemos as informações sobre o Projeto.

Para superar essas anomalias, precisamos normalizar os dados. Nas próximas seções, discutiremos sobre a normalização.

A chave primária de uma relação em um SGBD deve ser uma chave candidata, mas pode haver várias chaves candidatas para escolher. Quando se fala de normalização, é irrelevante qual chave é escolhida como chave primária.

1ª Forma Normal (1FN)

Definimos que uma tabela está na primeira forma normal, se e somente se, todas as colunas possuem um único valor, e não existam grupos repetitivos (colunas) em uma linha ou atributos compostos (Machado, 2014).

Para que uma tabela possa estar na 1FN, devemos seguir as seguintes regras:

1. Não devem existir colunas com dados repetidos ou similares;
2. Cada item de dados deve ser atômico (não possuir valores compostos);
3. Cada linha deve ser única, isto é, deve possuir uma chave primária;
4. Cada campo deve ter um nome exclusivo.

Importante

"Atômico" é o termo usado para descrever que um item de dados é único e indivisível.

Exemplo de dados repetidos:

Tabela 2

Tabela_Cliente				
Id_Cliente	Nome_Cliente	Telefone1	Telefone2	Telefone3
123	João da Silva	1234-2356	8945-5689	2563-8996

Neste exemplo, o banco de dados tenta armazenar números de telefones de contato para cada Cliente. O projetista criou três campos para manter números de telefone. Esse é um exemplo de "colunas com dados repetidos". Os números de telefone são o mesmo tipo de dados.

Exemplo de dados "não atômicos":

Tabela 3

Tabela_Cliente		
Id_Cliente	Nome_Cliente	E-mail
123	João da Silva	joao@gmail.com; joaosilva@gmail.com

Por mais que a tabela acima possua um campo de chave primária (Id_Cliente) e não existam dados repetidos, essa tabela não está na primeira forma normal (1FN) porque, como podemos notar, o cliente possui dois endereços de e-mail inseridos no campo "E-mail". Desse modo, os dados inseridos nesse campo não são atômicos.

Levando em consideração que existe a possibilidade de o cliente possuir mais que um (1) e-mail, a solução para esse cenário é criar uma nova entidade chamada E-mail e usar o campo chave (Id_Cliente) como chave estrangeira para fazer a ligação entre as duas entidades. Podemos observar o resultado nas tabelas a seguir:

Tabela 4

Tabela_Cliente	
Id_Cliente	Nome_Cliente
123	João da Silva
123	José Ferreira

Tabela 5

Tabela_Cliente		
Id_Email	Id_Cliente	E-mail
1	123	joao@gmail.com
2	123	joaosilva@gmail.com
3	123	josefer@gmail.com

Com essa solução, não há problemas quanto a inserir mais que um (1) e-mail por cliente e confere a fácil extração de dados de e-mail, pois, além de existir apenas uma coluna para e-mail, todos os dados são atômicos.

Podemos afirmar que essas tabelas estão na primeira forma normal (1NF), dado que obedecem às seguintes regras:

- Cada tabela tem uma chave primária;
- Cada nome de campo é exclusivo;
- Os dados são atômicos;
- Não possui campos repetidos / redundantes.

2ª Forma Normal (2FN)

Uma tabela está na segunda forma normal (2FN) se estiver na 1FN e não possuir campos que sejam funcionalmente dependentes de parte da chave primária (Machado, 2014).

As regras para a segunda forma normal são:

1. A tabela deve estar já na primeira forma normal (1FN);
2. Todos os atributos não-chave devem depender da chave primária completa, ou seja, não contenham dependência parcial.

Importante

Dependência parcial ocorre quando uma coluna depende apenas de parte de uma chave primária composta (Heuser, 2010).

A razão dessa regra é garantir que não sejam armazenados dados redundantes no banco de dados.

Vamos a um exemplo. Considere as seguintes tabelas:

Tabela 6

Tabela_Projeto	
Id_Projeto	Nome_Projeto
103	Manhattan
104	Houston

Tabela 7

Tabela_Empregado					
ID_ projeto	**ID_ Empregado**	**Nome Empregado**	**Cargo_ Empregado**	**Valor_Hora**	**Horas_ Trabalhadas**
103	1	João da Silva	Programador Sênor	30.00	50
103	2	Maria Fernanda	Gerente	80.00	50
104	1	João da Silva	Programador Sênior	30,00	20
104	3	Paulo Farias	Analista Sênior	40,00	10
104	4	Gustavo Fontes	Analista Sênior	40.00	20

No cenário acima, podemos perceber que a tabela **Projeto** está na segunda forma normal, pois todos os seus atributos dependem exclusivamente de sua chave primária. Contudo, a tabela **Empregado** não, posto que os atributos "Nome_Empregado", "Cargo_Empregado" e "Valor_Hora" dependem somente do campo chave "Id_ Empregado". São atributos que não estão ligados diretamente do campo chave "Id_ Projeto", ou seja, são atributos que não dependem da entidade **Projeto**. Por outro lado, o atributo "Horas_Trabalhadas" depende exclusivamente da chave composta da Tabela (Id_Projeto e Id_Empregado). A informação guardada nesse atributo permite saber quantas horas o empregado trabalhou no projeto específico.

Para que a tabela **Empregado** passe para a segunda forma normal (2FN), devemos dividir a tabela em duas novas tabelas, criando, assim, uma nova tabela chamada **Projeto_Horas_Trabalhadas**. Podemos conferir o resultado abaixo:

Tabela 8

Tabela_Projeto	
Id_Projeto	Nome_Projeto
103	Manhattan
104	Houston

Tabela_Projeto_Horas_Trabalhadas		
Id_Projeto	ID_Empregado	Horas_Trabalhdas
103	1	50
103	2	50
104	1	20
104	3	10

Podemos afirmar que as tabelas acima estão na segunda forma normal (2FN) porque todas as tabelas estão na 1FN e, além disso, os atributos não-chave de todas as tabelas possuem dependência exclusiva da chave primária de suas respectivas tabelas (sendo essas chaves primárias compostas ou não).

3ª Forma Normal (3FN)

Uma tabela encontra-se na terceira forma normal quando, além de estar na 2FN, não possua dependências transitivas.

Importante

Dependência transitiva ocorre quando uma coluna, além de depender da chave primária da tabela, depende de outra coluna ou conjunto de colunas da tabela (Heuser, 2010).

As regras para a terceira forma normal (3FN) são:

1. A tabela deve estar já na primeira forma normal (1FN);
2. Não existam atributos não-chave que dependam de outros atributos não-chave.

A razão dessa regra é detectar ainda outras fontes de dados redundantes. Se o valor de um atributo pode ser obtido simplesmente fazendo uso de outro atributo na tabela, então, ele não precisa estar na tabela. Criar uma tabela para armazenar esse tipo de atributo, tornará o banco de dados menor.

Desse modo, a tabela Empregado obtida na segunda forma normal (2FN), ainda possui dados redundantes. Perceba, nessa tabela, que o valor de hora de trabalho do empregado está vinculado ao seu cargo, ou seja, a informação do atributo "Valor_Hora" depende da informação contida no atributo "Cargo_Empregado".

Nesse contexto, para que a tabela Empregado passe para a terceira forma normal (3FN), devemos dividir a tabela em duas novas tabelas, criando, portanto, uma nova tabela chamada Cargo. Podemos conferir o resultado abaixo:

Tabela 9

Tabela_Empregado		
Id_Empregado	Nome_Empregado	Id_Cargo
1	João da Silva	1
2	Maria Fernanda	2
3	Paulo Farias	3
4	Gustavo Fontes	3

Tabela 10

Tabela_Cargo		
Id_Cargo	Cargo_Empregado	Valor_Hora
1	Programador Junior	30,00
2	Gerente	80,00
3	Analista Sênior	40,00

Tabela 11

Tabela_Projeto	
Id_Projeto	Nome_Projeto
103	Manhattan
104	Houston

Tabela 12

Tabela_Projeto		
Id_Projeto	Id_Empregado	Hora_Trabalhadas
103	1	50
103	2	50
104	1	20
104	3	10

Assim, podemos concluir que as tabelas acima estão na terceira forma normal (3FN), já que todas as tabelas estão na 2FN e, além disso, os atributos não-chave de todas as tabelas não possuem dependência transitiva de outros atributos não-chave em suas respectivas tabelas.

Forma Normal Boyce/Codd (FNBC) e 4ª e 5ª Forma Normal (4FN, 5FN)

Para Heuser (2010), a decomposição das tabelas até 3FN é suficiente para obter o esquema de um banco de dados. Contudo, na literatura, nos deparamos com a forma normal Boyce/Codd (FNBC) e 4ª e 5ª Forma Normal (4FN, 5FN).

Definimos que uma tabela está na forma normal Boyce/Codd (FNBC), se e somente se, cada determinante é uma chave candidata. A maioria das entidades em 3NF já estão em BCNF.

A tabela está na quarta forma normal (4FN) se nenhuma instância da tabela do banco de dados contiver dois ou mais dados independentes e multivalorados descrevendo a entidade relevante, então ela estará na quarta forma normal.

Uma tabela está na quinta forma normal (5FN) somente se estiver em 4NF e não puder ser decomposta em qualquer número de tabelas menores sem perda de dados.

Em Síntese

Explorar o Modelo Relacional é essencial para compreender a estrutura de armazenamento de dados em bancos de dados relacionais. Esse modelo se baseia na organização dos dados em tabelas, cada uma com um conjunto de tuplas, representando registros individuais, e atributos, descrevendo características específicas desses registros.

Definimos elementos cruciais desse modelo, como a chave primária, que identifica de forma única cada tupla em uma tabela, e a chave estrangeira, que estabelece relações entre diferentes tabelas. Esses conceitos são fundamentais para garantir a integridade e a consistência dos dados dentro do banco.

A normalização surge como um processo vital para estruturar os dados de forma eficiente e evitar redundâncias, minimizando possíveis problemas de inconsistência e permitindo um melhor desempenho do banco de dados. Ao seguir regras específicas de normalização, como a eliminação de dependências transitivas e a redução de redundâncias, é possível alcançar uma estrutura mais organizada e eficaz.

Em suma, o Modelo Relacional, com seus conceitos de tabelas, chaves, normalização e estruturação de dados, oferece um arcabouço robusto e bem definido para o armazenamento e gerenciamento de informações, garantindo maior consistência, confiabilidade e desempenho nos sistemas de banco de dados relacionais.

Atividades de Fixação

1 – Qual das seguintes declarações é verdadeira ao implementar um banco de dados relacional?

a. Um banco de dados relacional não permite a criação de chaves primárias.
b. Índices são usados apenas para fins de segurança em bancos de dados relacionais.
c. O SQL (*Structured Query Language*) é comumente usado para gerenciar bancos de dados relacionais.
d. O Modelo Relacional é focado, exclusivamente, na representação gráfica dos dados.
e. Bancos de dados relacionais não suportam transações multiusuário.

2 – Qual dos seguintes termos descreve melhor a relação entre tabelas em um banco de dados relacional?

a. Chave de criptografia.
b. Chave primária.
c. Marcador de dados.
d. Código-fonte.
e. Endereço de IP.

Atenção, estudante! Veja o gabarito desta atividade de fixação no fim deste conteúdo.

Material Complementar

Site

Normalização de Bancos de Dados Relacionais
https://goo.gl/F6ytEQ

Livro

PUGA, S.; FRANÇA, E.; GOYA, M. **Banco de Dados:** Implementação em SQL, PL SQL e Oracle 11g. 1.ed. São Paulo: Pearson Universidades, 2013.
Capítulo 5 – Normalização

Vídeos

Normalizacao: Primeira Forma Normal – 1FN
https://youtu.be/3kJKJNKiaD4

Normalizacao: Segunda Forma Normal – 2FN
https://youtu.be/mHoZZUYVFzk

Normalizacao: Terceira Forma Normal – 3FN
https://youtu.be/EZvrGEpyNbs

Referências

HEUSER, C. A. **Projeto de banco de dados**. 6.ed. Porto Alegre: Bookman, 2010.

MACHADO, F. N. R. **Banco de Dados**: projeto e implementação. 3. ed. São Paulo: Érica, 2014.

MELO, I. V. de A. Normalização de Banco de Dados Relacionais. **DSC UFCG**. maio, 2011. Disponível em: <http://www.dsc.ufcg.edu.br/~pet/jornal/maio2011/materias/recapitulando.html>. Acesso em:

NORMALIZAÇÃO de Banco de Dados: Primeira Forma Normal- 1FN. [S.l]. 01/05/2016. 1 vídeo (5 min.). Publicado pelo canal EdukaTI. Disponível em: <https://youtu.be/3kJKJNKia>. Acesso em: 18/01/2024.

NORMALIZAÇÃO de Banco de Dados: Segunda Forma Normal- 2FN. [S.l]. 01/05/2016. 1 vídeo (8 min.). Publicado pelo canal EdukaTI. Disponível em: <https://www.youtube.com/watch?v=mHoZZUYVFzk>. Acesso em: 18/01/2024.

NORMALIZAÇÃO de Banco de Dados: Terceira Forma Normal- 2FN. [S.l]. 01/05/2016. 1 vídeo (7 min.). Publicado pelo canal EdukaTI. Disponível em: <https://www.youtube.com/watch?v=EZvrGEpyNbs>. Acesso em: 18/01/2024.

PUGA, S.; FRANÇA, E.; GOYA, M. **Banco de Dados:** Implementação em SQL, PL SQL e Oracle 11g. 1.ed. São Paulo: Pearson Universidades, 2013.

Gabarito

Questão 1

c) O SQL (*Structured Query Language*) é comumente usado para gerenciar bancos de dados relacionais.

Justificativa: o SQL (*Structured Query Language*) é uma linguagem de consulta amplamente usada para gerenciar bancos de dados relacionais. Ela permite criar, consultar, modificar e manter bancos de dados relacionais. As outras opções estão incorretas: as chaves primárias são importantes em bancos de dados relacionais (a opção a está errada); os índices são usados para otimizar consultas, não apenas para fins de segurança (opção b está errada); o Modelo Relacional se concentra na estrutura dos dados, não em representações gráficas (opção d está errada); e os bancos de dados relacionais suportam transações multiusuário (opção e está errada).

Questão 2

b) Chave primária.

Justificativa: em um banco de dados relacional, a chave primária é o termo que descreve a coluna ou conjunto de colunas em uma tabela que identifica, exclusivamente, cada registro nessa tabela. A chave primária é fundamental para estabelecer relações entre tabelas, permitindo a vinculação de registros em uma tabela com registros em outra. As outras opções (a, c, d e e) não estão diretamente relacionadas à relação entre tabelas em um banco de dados relacional e, portanto, estão incorretas.