Projeto de banco de dados - modelagem lógica e física

O entendimento dos componentes do modelo relacional, formas normais e mapeamento conceitual-lógico, incluindo entidades, relacionamentos, atributos e especialização/generalização, é essencial para aplicar as regras no SGBD. A implementação eficiente do modelo no SGBD exige familiaridade com diretrizes físicas e é uma atividade rotineira para profissionais de banco de dados.

Profa. Nathielly de Souza Campos | Prof. Sidney Nicolau Venturi Filho

1. Itens iniciais

Propósito

É recomendável que você reproduza os exemplos práticos usando uma ferramenta para modelagem de dados. Certifique-se de ter baixado para seu computador a ferramenta livre BrModelo.

Objetivos

- Identificar os elementos do modelo relacional.
- Diferenciar formas normais.
- Aplicar o mapeamento conceitual-lógico.
- Identificar aspectos físicos para implementação do modelo no SGBD.

Introdução

Ao longo deste tema, vamos conhecer os principais conceitos e componentes do modelo relacional. Aprenderemos que o modelo relacional representa o banco de dados sob o formato de tabelas, estando presente em diversos sistemas gerenciadores de banco de dados (SGBD), tais como MySQL, Oracle, PostgreSQL e SQL Server.

Ainda, vamos estudar o processo de normalização como forma de avaliar a qualidade de um projeto de banco de dados relacional. Em seguida, conheceremos as regras que deverão ser aplicadas para obtermos um modelo lógico a partir de um modelo conceitual.

Finalmente, investigaremos alguns aspectos físicos que devem ser levados em consideração na implementação do modelo no SGBD.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Modelo relacional e componentes básicos de uma tabela

Neste conteúdo, vamos explorar os fundamentos para a organização estruturada de dados. Investigaremos os elementos-chave do modelo relacional, destacando a importância de tabelas, atributos e relacionamentos. Tal conhecimento é essencial para criar sistemas de banco de dados eficientes e compreender as bases que sustentam a gestão de informações em ambientes digitais. Prepare-se para conhecer o universo dos modelos relacionais e os componentes que moldam o armazenamento e a manipulação de dados.

Neste vídeo, você vai compreender o modelo relacional e os componentes básicos de uma tabela.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Modelo relacional

Relação é um termo usado na literatura formal da área de banco de dados. No contexto comercial, usa-se informalmente o termo tabela.

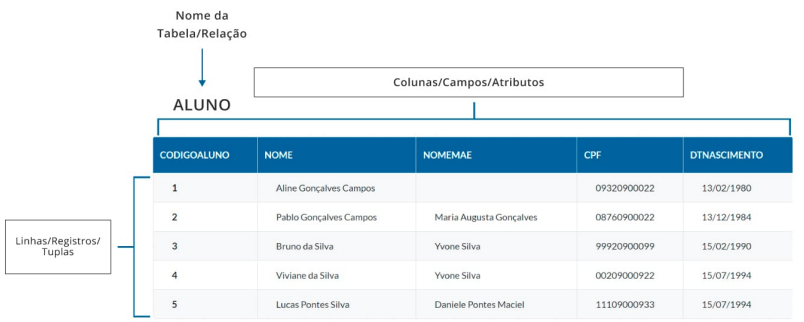
> O modelo relacional representa o banco de dados como uma coleção de relações.
>
> —
>
> (Elmasri; Navathe, 2019)

Componentes de uma tabela

Uma tabela corresponde a um conjunto não ordenado de linhas, que, na terminologia acadêmica, são conhecidas por tuplas.

As linhas de uma tabela são divididas em campos ou colunas, que, na academia, são chamados de atributos. Os campos são nomeados com objetivo de facilitar a interpretação dos dados armazenados.

Observe a tabela ALUNO a seguir:



CODIGOALUNO	NOME	NOMEMAE	CPF	DTNASCIMENTO
1	Aline Gonçalves Campos		09320900022	13/02/1980
2	Pablo Gonçalves Campos	Maria Augusta Gonçalves	08760900022	13/12/1984
3	Bruno da Silva	Yvone Silva	99920900099	15/02/1990
4	Viviane da Silva	Yvone Silva	00209000922	15/07/1994
5	Lucas Pontes Silva	Daniele Pontes Maciel	11109000933	15/07/1994

Componentes de uma tabela de banco de dados

Nome de tabelas

Em um banco de dados relacional, toda tabela deve possuir um nome único. Além disso, ao longo do nosso estudo, vamos perceber que a maioria das tabelas de um banco de dados representa entidades de um diagrama de entidade e relacionamento (DER).

Atenção

É importante que, na medida do possível, o nome da tabela represente com clareza o objeto modelado. Por exemplo, ao lermos o nome ALUNO, criamos a expectativa natural de que a tabela em questão armazene informações sobre alunos.

Colunas de tabelas

A primeira linha da tabela de exemplo contém os seguintes campos ou cabeçalhos de coluna:

- CODIGOALUNO
- NOME
- NOMEMAE
- CPF
- DTNASCIMENTO

Além disso, o nome de coluna deve ser único em cada tabela. Com isso, percebe-se que o nome de coluna ajuda a entender o papel ou a finalidade dela.

Exemplo

Podemos concluir que DTNASCIMENTO identifica a data de nascimento do aluno.

Outro ponto é que as colunas de uma tabela são monovaloradas, ou seja, é permitido manter no máximo um item de informação por vez. Por exemplo, é possível existir até uma ocorrência de data de nascimento na coluna DTNASCIMENTO.

As colunas de uma tabela possuem valores atômicos, ou seja, não admitem colunas compostas de outras. Por exemplo, não é possível subdividir CODIGODOALUNO em outros campos.

Ao implementar uma tabela em um banco de dados, é necessário definir um tipo de dado para cada coluna. Os mais comuns são: **caractere, numérico, data e booleano**.

> Alguns SGBDs (sistemas de gerência de banco de dados) permitem a definição do tipo de dados feita pelo usuário. Em uma linguagem mais técnica, o conjunto de valores que uma coluna pode assumir é denominado domínio da coluna ou domínio do campo.

Quando criarmos uma tabela, devemos definir se o valor da coluna é opcional ou obrigatório, em que especificar que uma coluna é opcional significa que os valores admitem vazio (NULL) ou nulo. Na tabela ALUNO, a coluna NOMEMAE da linha correspondente à aluna de código 1 está vazia.

Linhas de tabelas

As linhas da tabela, da segunda em diante, representam um item de informação cadastrado no banco de dados. Dizemos então que um item de informação corresponde a uma unidade básica que servirá para armazenamento e recuperação de dados. Isto é, se uma tabela de cadastro de alunos contém 10.000 linhas ou registros, podemos dizer que ela armazena dados de 10.000 alunos.

As linhas de uma tabela permitem o armazenamento de dados sempre de acordo com a semântica ou o significado do objeto. No caso em tela, a tabela armazena informações de ALUNOS. Portanto, queremos dizer que essa mesma tabela não deve ser utilizada para armazenar outros tipos de objetos, como disciplinas ou docentes.

Atividade 1

No contexto do banco de dados relacional, as tabelas são elementos fundamentais.

Qual é a principal função de uma tabela em um banco de dados relacional?

A

Ordenar registros alfanumericamente.

B

Armazenar arquivos binários.

C

Estruturar dados em linhas e colunas.

D

Controlar o acesso por senhas.

E

Realizar cálculos matemáticos.

A alternativa C está correta.

Uma tabela em um banco de dados relacional é uma estrutura organizada que armazena dados de maneira tabular, com linhas representando registros individuais e colunas representando atributos específicos. A opção correta, letra C: "Estruturar dados em linhas e colunas", destaca a função principal das tabelas ao organizar informações de forma eficiente e facilitar a consulta, a manipulação e o relacionamento entre os dados em um sistema de banco de dados relacional.

Chave primária

Vamos explorar a base da integridade e da identificação única em um banco de dados. A chave primária vai além de ser uma simples referência; é a âncora que define a singularidade de cada registro em uma tabela. Assim, vamos compreender a função da chave primária na estruturação de dados, garantindo coerência e

eficiência nas operações. Prepare-se para descobrir o que fundamenta a base da identificação e da organização precisa dos dados.

Neste vídeo, você vai compreender a chave primária como uma importante e fundamental restrição de integridade.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Em uma tabela, um SGBD precisa diferenciar uma linha das demais, isso é feito a partir da definição de uma restrição de integridade. Na prática, escolheremos uma ou mais coluna(s) para que seu(s) valor(es) se torne(m) únicos no banco de dados.

Atenção

Quando escolhemos uma coluna para ser chave primária, dizemos que estamos diante de uma chave simples. Se escolhermos mais de uma coluna, a chave é dita composta.

> Vamos estudar um exemplo de chave primária simples?

Para diferenciar um estudante dos demais na tabela ALUNO, podemos estabelecer a restrição de chave primária associada à coluna CODIGOALUNO. Ao fazermos isso, na prática, estamos delegando ao SGBD a responsabilidade de gerenciar essa restrição durante todo o ciclo de vida do banco de dados. Na tabela a seguir, deixamos em destaque a coluna CODIGOALUNO:

CODIGOALUNO	NOME	NOMEMAE	CPF
1	Aline Gonçalves Campos		09320900022
2	Pablo Gonçalves Campos	Maria Augusta Gonçalves	08760900022
3	Bruno da Silva	Yvone Silva	99920900099
4	Viviane da Silva	Yvone Silva	00209000922
5	Lucas Pontes Silva	Daniele Pontes Maciel	11109000933

ALUNO com destaque à coluna CODIGOALUNO, escolhida como chave primária simples
Nathielly de Souza Campos

Assim, podemos dizer que toda chave primária tem as seguintes propriedades:

Unicidade

O valor da chave primária não permite repetição.

Monovalorado

Toda linha da tabela possui no máximo um valor de chave primária.

Obrigatório

Toda linha da tabela necessariamente tem que ter um valor para a coluna que é chave primária. Em outras palavras, nenhum valor de chave primária deve ser vazio. Esta propriedade é conhecida por restrição de integridade de entidade.

Ao longo do nosso estudo, iremos aprender que alguns SGBDs permitem associar uma propriedade especial a um campo, denominada autoincremento. Por meio dessa propriedade, o SGBD incrementa automaticamente o valor de uma coluna quando um registro é adicionado à tabela. Trata-se de um mecanismo útil para gerar um valor único para cada registro.

Agora, vamos estudar um exemplo contendo chave primária composta.

Considere a tabela a seguir, que representa dados de dependentes de funcionários:

DEPENDENTE

CODIGOFUNCIONARIO	NRDEPENDENTE	NOME
1	1	Andrey Campos
1	2	Manoel Oliveira
2	1	João Silva
2	2	José Maciel

DEPENDENTE com destaque às colunas CODIGOFUNCIONARIO, NRDEPENDENTE escolhida como chave primária composta
Nathielly de Souza Campos

A tabela possui uma chave primária composta pelo par de colunas CODIGOFUNCIONARIO, NRDEPENDENTE. Devemos notar que nenhuma das colunas que compõem a chave é suficiente para, isoladamente, diferenciar uma linha das demais, visto que:

- Um CODIGOFUNCIONARIO pode aparecer em diferentes linhas da tabela;
- Um NRDEPENDENTE pode aparecer em diferentes linhas da tabela.

Chave mínima

Uma chave primária deve ser mínima. Com isso, queremos dizer que todas as colunas que a formam devem ser necessárias e suficientes para diferenciar uma linha das demais na tabela. Outro ponto importante é que uma chave mínima não diz respeito ao quantitativo de colunas que a forma.

Chave primária simples

A chave primária simples (coluna CODIGOALUNO) da tabela ALUNO é mínima. Cada valor de CODIGOALUNO é suficiente para diferenciar um aluno dos demais. Perceba que, se decidíssemos definir o par de colunas CODIGOALUNO, CPF como chave primária composta para a tabela ALUNO, a chave não seria mínima, visto que CODIGOALUNO por si só é suficiente para diferenciar um estudante dos outros.

Chave primária composta

A chave primária composta (CODIGOFUNCIONARIO, NRDEPENDENTE) da tabela DEPENDENTE é mínima. Somente a coluna CODIGOFUNCIONARIO não diferencia um dependente dos demais, pois um funcionário pode ter diversos dependentes. De modo semelhante, somente a coluna NRDEPENDENTE não diferencia um dependente dos outros, dado que o valor dela aparece em mais de uma linha da tabela DEPENDENTE.

Chave candidata

Ao projetarmos uma tabela, pode ser que mais de uma coluna sirva para diferenciar uma linha das demais. Por exemplo, na tabela ALUNO, tanto CODIGOALUNO quanto CPF poderiam ser utilizados como chave primária. Logo, podemos dizer que CODIGOALUNO e CPF são chaves candidatas.

Chave alternativa

A partir do momento em que escolhemos CODIGOALUNO para ser a chave primária da tabela ALUNO, passamos a considerar CPF como uma chave alternativa.

Alguns desenvolvedores preferem escolher para chave primária uma coluna *artificial*, ou seja, que não tenha dependência das colunas criadas, com objetivo de manter alguma informação referente ao negócio sendo modelado.

Atividade 2

A função da chave primária é central no banco de dados relacional.

Qual opção define corretamente o papel essencial dessa chave?

A

Facilitar a ordenação dos registros.

B

Fornecer uma cópia de backup dos dados.

C

Garantir a unicidade e a identificação exclusiva de cada registro.

D

Definir a formatação visual das tabelas.

E

Controlar o acesso aos dados por meio de senhas.

A alternativa C está correta.

A chave primária tem um papel importante ao assegurar que todo registro em uma tabela seja único e tenha uma identificação exclusiva. Isso promove a integridade referencial no banco de dados, evitando duplicatas e estabelecendo uma base sólida para relacionamentos entre tabelas. Essa característica é essencial para manter a consistência e a eficácia das operações no banco de dados relacional.

Chave estrangeira

Ao falarmos de chave estrangeira, tratamos da interconexão inteligente entre tabelas em um banco de dados. A chave estrangeira estabelece relações significativas, conectando dados de maneira coesa. Assim, vamos explorar como essa chave especial transcende fronteiras de tabelas, viabilizando integridade referencial e enriquecendo a compreensão dos dados. Este é um convite para conhecer a importância da chave estrangeira na harmonização e na consistência das informações, realizando a coesão em sistemas de banco de dados.

Neste vídeo, você vai compreender a chave estrangeira como garantidora da restrição de integridade referencial.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Um banco de dados relacional é composto por um conjunto de tabelas. Na maioria dos casos, existe algum tipo de relacionamento entre elas. O relacionamento entre tabelas é um dos conceitos fundamentais em projeto de banco de dados relacionais. Entenda os seguintes conceitos:

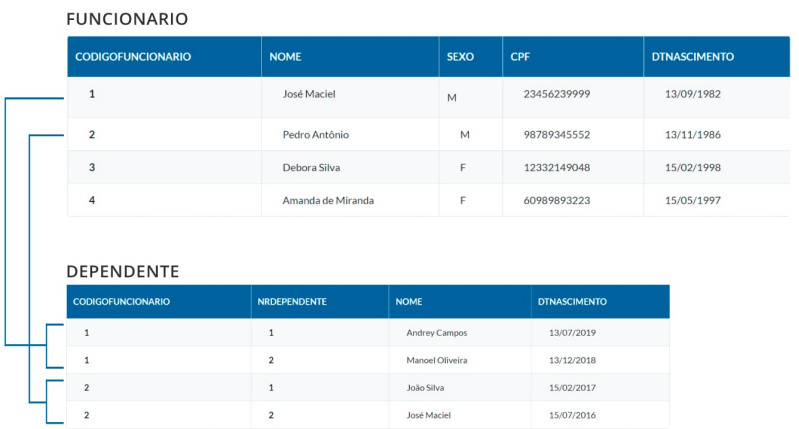
Modelo relacional

Decorre do termo relação, função matemática que conhecemos popularmente como tabela.

Relacionamento

É o relacionamento que existe entre as tabelas.

Em geral, uma empresa possui informações sobre funcionários e seus dependentes. Vamos observar a tabela a seguir:



FUNCIONARIO

CODIGOFUNCIONARIO	NOME	SEXO	CPF	DTNASCIMENTO
1	José Maciel	M	23456239999	13/09/1982
2	Pedro Antônio	M	98789345552	13/11/1986
3	Debora Silva	F	12332149048	15/02/1998
4	Amanda de Miranda	F	60989893223	15/05/1997

DEPENDENTE

CODIGOFUNCIONARIO	NRDEPENDENTE	NOME	DTNASCIMENTO
1	1	Andrey Campos	13/07/2019
1	2	Manoel Oliveira	13/12/2018
2	1	João Silva	15/02/2017
2	2	José Maciel	15/07/2016

Relacionamento entre FUNCIONARIO e DEPENDENTE

A tabela permite as seguintes interpretações:

- Todo dependente está associado a um funcionário. Por exemplo, o valor (1) de CODIGOFUNCIONARIO na tabela DEPENDENTE permite identificar o funcionário responsável, neste caso José Maciel;
- Um funcionário pode ter diversos dependentes. Por exemplo, o valor (2) de CODIGOFUNCIONARIO nas duas últimas linhas da tabela DEPENDENTE permite concluir que o funcionário Pedro Antônio possui dois dependentes;
- Um funcionário pode não ter dependentes. Por exemplo, o valor (3) de CODIGOFUNCIONARIO na tabela FUNCIONARIO não aparece na tabela DEPENDENTE.

No exemplo, CODIGOFUNCIONARIO da tabela DEPENDENTE é chave estrangeira, pois todo valor dessa coluna necessariamente aparece como valor da coluna CODIGOFUNCIONARIO, chave primária de FUNCIONARIO. Como consequência, podemos concluir que todo dependente está vinculado a um funcionário.

> Uma chave estrangeira é uma coluna ou uma combinação de colunas, cujos valores aparecem necessariamente na chave primária de uma tabela.
>
> —
>
> (Heuser, 2009, p. 34)

Deve-se notar que, mesmo que a coluna CODIGOFUNCIONARIO de DEPENDENTE tenha o mesmo nome da coluna que é chave primária em FUNCIONARIO, é possível nomeá-la de forma distinta. No entanto, usar o mesmo nome facilita na identificação das colunas relacionadas.

Restrições impostas pela chave estrangeira

Percebemos que a chave estrangeira serve para implementar relacionamentos entre tabelas. Para que o banco de dados permaneça íntegro, algumas restrições devem ser controladas e obedecidas pelo SGBD:

- Inclusão de linha na tabela que possui chave estrangeira: o sistema deve garantir que o valor da chave estrangeira exista como valor da coluna da chave primária referenciada. Em nosso exemplo, ao incluir um dependente, a coluna CODIGOFUNCIONARIO só pode assumir um dos valores (1,2,3,4).
- Alteração de valor da chave estrangeira: o sistema deve garantir que o novo valor da chave estrangeira exista como valor de coluna da chave primária referenciada. Em nosso exemplo, ao alterar um responsável de algum dependente, a coluna CODIGOFUNCIONARIO só pode assumir um dos valores (1,2,3,4).
- Exclusão de linha em tabela que contém chave primária referenciada pela chave estrangeira: o sistema deve garantir que todo valor de chave estrangeira sempre faça referência para o valor de alguma chave primária. Em nosso exemplo, os funcionários que têm código 1 ou 2 não devem ser excluídos, pois possuem vínculo com a tabela de dependentes.
- Alteração de valor da chave primária referenciada pela chave estrangeira: o sistema deve garantir que o novo valor da chave primária da tabela principal seja replicado nas respectivas dependências. Em nosso exemplo, se alterarmos o valor de CODIGOFUNCIONARIO (tabela FUNCIONARIO) de 2 para 5, o sistema deve garantir a propagação da atualização nas duas últimas linhas de DEPENDENTE.
- Os exemplos estudados representam o que conhecemos por integridade referencial, ou seja, os valores de chave estrangeira devem aparecer na chave primária da tabela referenciada.

Atividade 3

A respeito dos tipos de chaves em projeto de banco de dados relacional, analise as proposições a seguir:

I. Quando mais de uma coluna servir para diferenciar uma linha das demais em uma tabela relacional e uma delas é escolhida como chave primária, as restantes são denominadas chaves alternativas.
II. Uma chave primária corresponde a uma coluna ou uma combinação de colunas cujos valores servem para diferenciar uma linha das demais de uma tabela.
III. A chave estrangeira permite a implementação de relacionamentos em um banco de dados relacional.

Assinale a alternativa verdadeira:

A

Somente a proposição I é correta.

B

Somente a proposição II é correta.

C

Somente a proposição IIII é correta.

D

Todas as proposições são corretas.

E

Somente as proposições I e II são corretas.

A alternativa D está correta.

De fato, i) quando diversas colunas são candidatas à chave primária e uma é escolhida, as demais passam a ser denominadas chaves alternativas; ii) o objetivo da chave primária é tornar única cada linha de uma tabela do banco de dados; iii) as chaves estrangeiras representam restrições capazes de representar relacionamentos entre tabelas.

Esquema diagramático de banco de dados relacional

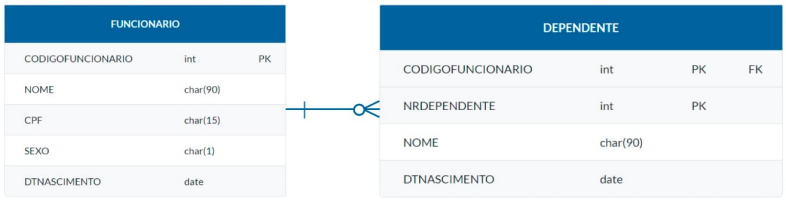
Ao visualizarmos o esquema diagramático de um banco de dados relacional, percebemos a arquitetura visual que traduz a complexidade organizacional dos dados. Esse esquema é mais do que uma representação gráfica; é um mapa que revela interconexões entre tabelas, atributos e chaves. Vamos explorar as linhas e as formas que delineiam relacionamentos, proporcionando uma compreensão visual clara da estrutura subjacente.

Neste vídeo, você vai compreender a importância do esquema diagramático de um banco de dados relacional, que traduz a complexidade organizacional dos dados.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Diversas ferramentas de modelagem permitem o uso de alguma notação gráfica para representar um banco de dados relacional. Na imagem a seguir, temos um diagrama que foi construído a partir de uma ferramenta comercial.



Esquema diagramático “pé de galinha” envolvendo as tabelas FUNCIONARIO e DEPENDENTE.Nathielly de Souza Campos

A notação utilizada é conhecida por pé de galinha (inglês *Crow's Foot*). Nesse esquema diagramático:

- Cada tabela é representada por um retângulo com duas divisões;
- Na primeira subdivisão, adicionamos o nome da tabela;
- Na segunda subdivisão, aparecem as colunas da tabela, com as informações sobre o nome, o tipo de dados, além de símbolos representativos de chave primária PK (do Inglês, Primary Key) e chave estrangeira FK (do Inglês, Foreign Key);
- O símbolo “colado” na tabela DEPENDENTE, semelhante a um pé de galinha, representa o lado N do relacionamento entre as tabelas.

Esquema textual de banco de dados relacional

O banco de dados mostrado no diagrama anterior, pode ser declarado sob o formato textual, conforme exemplo a seguir:

```
FUNCIONARIO (CODIGOFUNCIONARIO, NOME, CPF, SEXO, DTNASCIMENTO)
 DEPENDENTE (CODIGOFUNCIONARIO, NRDEPENDENTE, NOME, DTNASCIMENTO)
  CODIGOFUNCIONARIO REFERENCIA FUNCIONARIO
```

Observe que, no esquema textual, as tabelas são declaradas com informações sobre o nome, além de uma lista contendo as suas respectivas colunas. As chaves primárias são sublinhadas. Por fim, as chaves estrangeiras são declaradas usando o padrão nomecoluna(s) referencia nometabela(s).

Neste módulo, estudamos os principais elementos do modelo relacional de banco de dados.

Atividade 4

Considere que em uma instituição de ensino superior (IES) exista um banco de dados relacional denominado BDIES que possui a tabela ALUNO cujos campos estão assim descritos:

ALUNO		
CODIGOALUNO	int	PK
NOME	char(90)	
SEXO	char(1)	
DTNASCIMENTO	date	

Tabela aluno

De acordo com os fundamentos do modelo relacional, é correto afirmar que:

A

O banco de dados BDIES pode possuir uma coleção de tabelas, todas com nome exclusivo, com colunas que podem conter valores dentro de um domínio.

B

A coluna CODIGOALUNO é chave estrangeira na tabela ALUNO.

C

A coluna CODIGOALUNO pode ser opcional.

D

A coluna NOME é multivalorada.

E

A coluna DTNASCIMENTO é uma chave alternativa da tabela.

A alternativa A está correta.

De fato, toda tabela de um banco de dados relacional tem que possuir um nome único. As colunas de uma tabela devem estar associadas a um domínio ou tipo de dados.

Normalização

Neste estudo, vamos organizar informações de maneira eficiente. Tal processo, fundamental em bancos de dados, visa reduzir redundâncias e anomalias, proporcionando consistência e integridade. Ao compreender os princípios da normalização, desvendamos a chave para estruturas de dados mais flexíveis e otimizadas, capacitando sistemas a lidar com informações de maneira eficaz e escalável. Prepare-se para descobrir a essência da normalização e os segredos da organização inteligente de dados.

Neste vídeo, você vai compreender a importância da normalização na melhoria da consistência e da integridade dos dados em um banco de dados relacional, reduzindo redundâncias e anomalias.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Ao longo da nossa jornada, conhecemos os principais elementos do modelo relacional. Quando trabalhamos com modelagem das tabelas de um banco de dados, ao criarmos as tabelas, é natural definirmos colunas que têm relação com as características do objeto sendo modelado.

No entanto, modelar um banco de dados relacional não se resume simplesmente a usar uma **ferramenta CASE** e adicionar tabelas e relacionamentos sem que haja algum critério para essa construção.

Ferramenta CASE

Software de apoio ao desenvolvimento de sistemas, desde a análise e modelagem até a programação e testes.

> Ao longo deste módulo, estudaremos o assunto normalização, que ajudará a responder se um banco de dados foi bem projetado. Além disso, é possível executar o processo de normalização a partir de qualquer representação de dados. Isso significa que podemos iniciar o processo a partir de uma tela de sistema, ou mesmo um relatório.

A normalização é um processo baseado no conceito de forma normal (FN), que pode ser vista como uma regra, a qual deve ser observada na semântica de uma tabela, para que a considerem bem projetada.

Atenção

Na literatura de banco de dados, há diversas formas normais: 1FN, 2FN, 3FN, FNBC, 4FN e 5FN. No entanto, para fins práticos, no contexto da maior parte dos projetos de banco de dados relacionais, costumamos executar o processo de normalização até a 3FN.

Dividiremos o processo de normalização até a 3FN de acordo com o seguinte roteiro:

- Identificar a origem dos dados.
- Construir tabela não normalizada a partir dos dados.

- Aplicar as regras da primeira forma normal (1FN).
- Aplicar as regras da segunda forma normal (2FN).
- Aplicar as regras da terceira forma normal (3FN).

Considere o relatório expresso na tabela, que informa os docentes participantes em projetos de pesquisa de uma instituição de ensino superior (IES):

RELATÓRIOS DE ALOCAÇÃO DOCENTE A PROJETOS DE PESQUISA					
CÓDIGO DO PROJETO: PRODATA			TIPO: ANÁLISE DE DADOS		
DESCRIÇÃO: DESENVOLVIMENTO DE AMBIENTE PARA ANÁLISE DE DADOS					
CÓDIGO DO DOCENTE	NOME	CATEGORIA	SALÁRIO	DATA DE INÍCIO	TEMPO ALOCADO
DOC001	JOSÉ	ADJUNTO	R$ 6.000,00	01/02/2019	16
DOC002	LUCIANO	TITULAR	R$ 16.000,00	01/02/2020	4
DOC003	GILSON	ADJUNTO	R$ 6.000,00	01/02/2019	16
DOC004	MARTA	TITULAR	R$ 16.000,00	01/02/2020	4

CÓDIGO DO PROJETO: PROMED			TIPO: ANÁLISE CLÍNICA		
DESCRIÇÃO: ATENDIMENTO COMUNITÁRIO E VACINAÇÃO					
CÓDIGO DO DOCENTE	NOME	CATEGORIA	SALÁRIO	DATA DE INÍCIO	TEMPO ALOCADO
DOC001	JOSÉ	ADJUNTO	R$ 6.000,00	01/02/2019	16
DOC010	MARIA	ADJUNTO	R$ 6.000,00	01/06/2020	0
DOC004	MARTA	TITULAR	R$ 16.000,00	01/05/2020	1

Alocação de docentes a projetos de pesquisa em uma IES
Nathielly de Souza Campos

De acordo com o relatório, nós podemos perceber que:

- Os projetos são caracterizados por código, descrição e categoria (tipo).
- Para cada docente alocado em projeto, aparecem os seguintes campos: código e nome do docente, sua categoria e salário, além da data de início de atuação no projeto, bem como o tempo alocado.

Agora nós executaremos o segundo passo do roteiro, que corresponde a criar tabela não normalizada a partir do relatório.

Tabela não normalizada

A tabela a seguir representa uma tabela não normalizada, denominada PROJETO, criada a partir do relatório:

CODIGOPROJETO	TIPO	DESCRICAO	DOCENTE			
			CODIGODOCENTE	NOME	CATEGORIA	SALARIO
PRODATA	ANÁLISE DE DADOS	DESENVOLVIMENTO DE AMBIENTE PARA ANÁLISE DE DADOS	DOC001	JOSÉ	ADJUNTO	R$ 6.000,00
			DOC002	LUCIANO	TITULAR	R$ 16.000,00
			DOC003	GILSON	ADJUNTO	R$ 6.000,00
			DOC004	MARTA	TITULAR	R$ 16.000,00
PROMED	ANÁLISE CLÍNICA	ATENDIMENTO COMUNITÁRIO E VACINAÇÃO	DOC001	JOSÉ	ADJUNTO	R$ 6.000,00
			DOC010	MARIA	ADJUNTO	R$ 6.000,00
			DOC004	MARTA	TITULAR	R$ 16.000,00

Representação dos dados do relatório em tabela não normalizada
Nathielly de Souza Campos

A representação textual da tabela está expressa a seguir:

> PROJETO (CODIGOPROJETO, TIPO, DESCRICAO, (CODIGODOCENTE, NOME, CATEGORIA, SALARIO, DATAINICIO, TEMPOMESES))

Nessa representação, a coluna CODIGOPROJETO diferencia cada projeto dos demais. A coluna CODIGODOCENTE diferencia os docentes alocados no contexto de um projeto.

Agora, observe com atenção a lista representativa da tabela não normalizada:

- Cada linha da tabela não normalizada representa a informação de alocação de um docente a um projeto de pesquisa.
- Note que a coluna DOCENTE é composta por um conjunto de colunas:CODIGODOCENTE, NOME, CATEGORIA, SALARIO, DATAINICIO e TEMPOMESES. Estamos diante de uma coluna composta.
- Perceba, também, que se olharmos isoladamente para os valores das colunas que compõem a coluna DOCENTE, vamos perceber repetição. Isso acontece no caso dos funcionários José e Marta. A mesma informação está representada mais de uma vez, o que representa redundância.

Agora que construímos a tabela não normalizada, vamos à próxima etapa, que terá como saída um conjunto de tabelas na 1FN.

Atividade 1

No universo da normalização de dados, os conceitos fundamentais moldam a eficiência e a integridade nos bancos de dados.

A

Aumentar a redundância dos dados.

B

Simplificar a estrutura do banco de dados.

C

Introduzir dados fictícios para teste.

D

Permitir o acesso irrestrito aos dados.

E

Melhorar a consistência e a integridade dos dados.

A alternativa E está correta.

O principal objetivo da normalização é realizar melhorias na consistência e na integridade dos dados em um banco de dados relacional, reduzindo redundâncias e anomalias. Esse processo visa criar estruturas mais eficientes e flexíveis para manipulação de informações.

Primeira Forma Normal (1FN)

Ao explorarmos o conteúdo da Primeira Forma Normal (1NF), vamos conhecer as bases da organização de dados. Essa etapa da normalização em bancos de dados visa eliminar repetições de dados, promovendo a uniformidade e a estruturação. Compreender os princípios da 1NF é essencial para criar sistemas robustos, garantindo que todo atributo em uma tabela contenha valores atômicos, liberando caminho para estruturas mais eficientes e facilitando operações de consulta e manipulação de dados.

Neste vídeo, você vai compreender a importância da Primeira Forma Normal (1FN) na normalização.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Uma tabela está de acordo com a 1FN quando não possui atributo(s) multivalorado(s) nem atributo(s) composto(s).

Devemos recordar que, como resultado da etapa anterior, foi criada tabela não normalizada. Para ficar de acordo com a 1FN, nós executaremos os passos a seguir:

Passo 1

Criar tabela na 1FN com a mesma chave primária da tabela não normalizada, além das colunas atômicas da própria tabela não normalizada.

Passo 2

Criar uma tabela na 1FN para cada coluna composta, identificada na tabela não normalizada. Cada tabela terá uma chave primária composta pela chave primária da tabela criada no passo anterior e pela coluna identificada como composta. Além disso, terá as colunas membro da coluna em questão.

Passo 3

Criar uma tabela na 1FN para cada coluna multivalorada. Cada tabela terá uma chave primária composta pela chave primária da tabela não normalizada e pela coluna identificada como multivalorada.

Ao aplicarmos os passos à tabela PROJETO, teremos como resultado as seguintes tabelas em 1FN:

PROJETO (CODIGOPROJETO, TIPO, DESCRICAO)
PROJETODOCENTE (CODIGOPROJETO, CODIGODOCENTE, NOME, CATEGORIA, SALARIO, DATAINICIO, TEMPOMESES)

Ainda, de acordo com a representação textual, devemos perceber que:

- A coluna CODIGOPROJETO da tabela PROJETO diferencia um projeto dos demais.
- As colunas CODIGOPROJETO, CODIGODOCENTE compõem a chave primária da tabela PROJETODOCENTE, visto que um mesmo docente pode atuar em diversos projetos.

A seguir, veja a representação do conteúdo das tabelas, com base nos dados originalmente expressos no relatório de alocação docente a projetos:

PROJETO

CODIGOPROJETO	TIPO	DESCRICAO
PRODATA	ANÁLISE DE DADOS	DESENVOLVIMENTO DE AMBIENTE PARA ANÁLISE DE DADOS
PROMED	ANÁLISE CLÍNICA	ATENDIMENTO COMUNITÁRIO E VACINAÇÃO

Tabela: Representação dos dados do relatório em tabelas na 1FN.
Nathielly de Souza Campos.

PROJETODOCENTE

CODIGOPROJETO	CODIGODOCENTE	NOME	CATEGORIA	SALARIO	DATAINICIO	TEMPOMESES

PRODATA	DOC001	JOSÉ	ADJUNTO	R$ 6.000,00	01/02/2019	16
PRODATA	DOC002	LUCIANO	TITULAR	R$ 16.000,00	01/02/2020	4
PRODATA	DOC003	GILSON	ADJUNTO	R$ 6.000,00	01/02/2019	16
PRODATA	DOC004	MARTA	TITULAR	R$ 16.000,00	01/02/2020	4
PROMED	DOC001	JOSÉ	ADJUNTO	R$ 6.000,00	01/02/2019	16
PROMED	DOC010	MARIA	ADJUNTO	R$ 6.000,00	01/06/2020	0
PROMED	DOC004	MARTA	TITULAR	R$ 16.000,00	01/05/2020	1

Representação dos dados do relatório em tabelas na 1FN
Nathielly de Souza Campos

Dependência funcional

Se observarmos os dados da tabela PROJETODOCENTE, vamos concluir que o nome do docente é o mesmo para cada código de docente. Parece então existir uma relação de dependência entre as colunas NOME e CODIGODOCENTE.

Assim, podemos expressar essa relação da seguinte maneira:

CODIGODOCENTE → NOME.

Com isso, dizemos que a coluna CODIGODOCENTE é determinante da coluna NOME.

Podemos dizer também que NOME é dependente de CODIGODOCENTE, isto é, determinado por CODIGODOCENTE.

⚠

Atenção

A generalização dessa informação representa o conceito de dependência funcional. Dessa forma, seja X um conjunto de atributos e Y um atributo; X → Y significa que o conjunto de atributos X determina o atributo Y.

Dependência funcional parcial

Observe novamente a tabela PROJETODOCENTE:

PROJETODOCENTE

CODIGOPROJETO	CODIGODOCENTE	NOME	CATEGORIA	SALARIO	DATAINICIO	TEMPOMESES

PRODATA	DOC001	JOSÉ	ADJUNTO	R$ 6.000,00	01/02/2019	16
PRODATA	DOC002	LUCIANO	TITULAR	R$ 16.000,00	01/02/2020	4
PRODATA	DOC003	GILSON	ADJUNTO	R$ 6.000,00	01/02/2019	16
PRODATA	DOC004	MARTA	TITULAR	R$ 16.000,00	01/02/2020	4
PROMED	DOC001	JOSÉ	ADJUNTO	R$ 6.000,00	01/02/2019	16
PROMED	DOC010	MARIA	ADJUNTO	R$ 6.000,00	01/06/2020	0
PROMED	DOC004	MARTA	TITULAR	R$ 16.000,00	01/05/2020	1

Representação dos dados da tabela PROJETODOCENTE
Nathielly de Souza Campos

Se analisarmos a relação CODIGOPROJETO, CODIGODOCENTE → NOME, iremos perceber que NOME é dependente somente de CODIGODOCENTE, ou seja, não é necessária a existência do par CODIGOPROJETO, CODIGODOCENTE para determinar o nome do docente. Estamos diante de uma dependência funcional parcial, visto que identificamos uma coluna dependente somente de parte da chave primária composta.

Atividade 2

Na normalização em bancos de dados, a Primeira Forma Normal (1NF) contribui para a organização eficiente das informações.

Qual princípio define a Primeira Forma Normal?

A

Eliminação de redundâncias.

B

Restrição à atomicidade dos valores.

C

Hierarquia de dados complexa.

D

Ausência de relacionamentos entre tabelas.

E

Ênfase na diversidade de tipos de dados.

A alternativa B está correta.

A primeira Forma Normal (1NF) requer que cada atributo em uma tabela contenha valores atômicos, não permitindo a presença de estruturas complexas. Isso assegura a simplicidade e a uniformidade dos dados, contribuindo para a eficiência e a integridade na organização das informações em um banco de dados.

Segunda Forma Normal (2FN)

Ao tratarmos da Segunda Forma Normal (2FN), vamos explorar os alicerces da normalização em bancos de dados. Essa segunda etapa crucial da normalização visa aprimorar a estruturação dos dados, eliminando dependências parciais e garantindo eficiência. Compreender os princípios da 2FN é essencial para a construção de sistemas de banco de dados mais robustos, nos quais as tabelas refletem, com precisão, as relações entre entidades, proporcionando integridade e facilitando operações avançadas de consulta e de manipulação.

Neste vídeo, você vai compreender a importância da Segunda Forma Normal (2FN) na normalização.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

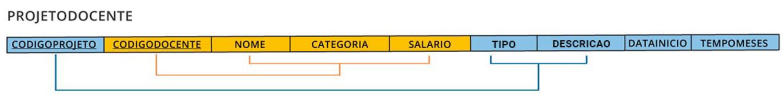
Uma tabela está na 2FN, caso esteja na 1FN e não haja dependências funcionais parciais.

Se analisarmos com um pouco mais de atenção cada coluna não chave da tabela PROJETODOCENTE, iremos perceber que, além da coluna NOME, as colunas CATEGORIA e SALARIO também só dependem da coluna CODIGODOCENTE.

Para ficar de acordo com a 2FN, será necessário eliminarmos as dependências parciais, conforme os passos a seguir:

- Manter no modelo cada tabela que possua chave primária simples.
- Identificar cada dependência parcial.
- Criar uma tabela para cada dependência parcial identificada.

A imagem a seguir identifica as colunas dependentes de parte da chave primária na tabela PROJETODOCENTE:



Representação das dependências parciais na tabela PROJETODOCENTE

Ao aplicarmos os passos ao modelo, teremos como resultado as seguintes tabelas na 2FN:

PROJETO (CODIGOPROJETO, TIPO, DESCRICAO)
PROJETODOCENTE (CODIGOPROJETO, CODIGODOCENTE, DATAINICIO,TEMPOMESES)
DOCENTE (CODIGODOCENTE, NOME, CATEGORIA, SALARIO)

A seguir, veremos a representação do conteúdo das tabelas na 2FN, com base nos dados, originalmente expressos no relatório de alocação docente a projetos:

PROJETO

CODIGOPROJETO	TIPO	DESCRICAO
PRODATA	ANÁLISE DE DADOS	DESENVOLVIMENTO DE AMBIENTE PARA ANÁLISE
PROMED	ANÁLISE CLÍNICA	ATENDIMENTO COMUNITÁRIO E VACINAÇÃO

PROJETODOCENTE

CODIGOPROJETO	CODIGODOCENTE	DATAINICIO
PRODATA	DOC001	01/02/2019
PRODATA	DOC002	01/02/2020
PRODATA	DOC003	01/02/2019
PRODATA	DOC004	01/02/2020
PROMED	DOC001	01/02/2019
PROMED	DOC010	01/06/2020
PROMED	DOC004	01/05/2020

DOCENTE

CODIGODOCENTE	NOME	CATEGORIA	SALARIO
DOC001	JOSÉ	ADJUNTO	R$ 6.000,00
DOC002	LUCIANO	TITULAR	R$ 16.000,00
DOC003	GILSON	ADJUNTO	R$ 6.000,00
DOC004	MARTA	TITULAR	R$ 16.000,00
DOC010	MARIA	ADJUNTO	R$ 6.000,00

Representação dos dados do relatório em tabelas em 2FN
Nathielly de Souza Campos

O modelo está na 2FN, pois, além de estar na 1FN, não existem dependências parciais.

Atividade 3

Na normalização de bancos de dados, o que caracteriza um esquema de banco de dados que atenda à Segunda Forma Normal (2FN)?

A

Ausência de dados duplicados.

B

Dependência total da chave primária.

C

Eliminação de dependências parciais.

D

Atributos armazenados de forma não atômica.

E

Nenhum relacionamento entre tabelas.

A alternativa C está correta.

A Segunda Forma Normal (2FN) é alcançada quando uma tabela está na 1FN e não possui dependências parciais, ou seja, cada atributo não chave depende apenas da chave primária, eliminando complexidades desnecessárias na estrutura de dados.

Terceira Forma Normal (3FN)

Ao explorar o conteúdo da Terceira Forma Normal (3FN), vamos conhecer as nuances da normalização em bancos de dados. Essa etapa visa à eliminação de dependências transitivas, promovendo a integridade e a eficiência estruturais. Compreender os princípios da 3FN auxilia na construção de sistemas de bancos de dados mais refinados, em que todo atributo está diretamente relacionado à chave primária, evitando redundâncias e facilitando a manutenção e a consulta de dados.

Neste vídeo, você vai compreender a importância da Terceira Forma Normal (3FN) no contexto da normalização.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Dependência funcional transitiva

Se observarmos os dados da tabela DOCENTE gerada na etapa anterior, podemos concluir que o valor do salário é o mesmo em cada categoria. Perceba que parece então existir uma relação de dependência entre as colunas CATEGORIA e SALARIO. Assim, podemos expressar essa relação da seguinte maneira:

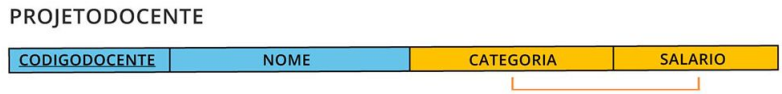
CATEGORIA → SALARIO.

Com isso, dizemos que a coluna CATEGORIA é determinante da coluna SALARIO. Podemos dizer também que SALARIO é dependente de CATEGORIA.

> ⚠
>
> Atenção
>
> Estamos diante de um exemplo de dependência funcional, em que o determinante é uma coluna que não pertence à chave primária da tabela. Uma dependência funcional transitiva ocorre quando uma coluna é dependente de alguma coluna não-chave da tabela.

Uma tabela está em 3FN caso esteja na 2FN e não possua dependências transitivas.

A imagem a seguir identifica a dependência transitiva na tabela PROJETODOCENTE:



Representação da dependência transitiva na tabela PROJETODOCENTE

Para ficar de acordo com a 3FN, será necessário eliminarmos as dependências transitivas, conforme os passos a seguir:

- Manter no modelo cada tabela que tenha menos de duas colunas não chave.
- Identificar cada dependência transitiva.
- Criar uma tabela para cada dependência transitiva identificada.

Ao aplicarmos os passos ao modelo, teremos como resultado as seguintes tabelas na 3FN:

PROJETO (CODIGOPROJETO, TIPO, DESCRICAO)
PROJETODOCENTE (CODIGOPROJETO, CODIGODOCENTE, DATAINICIO, TEMPOMESES)
DOCENTE (CODIGODOCENTE, NOME, CATEGORIA)
CATEGORIA (CATEGORIA, SALARIO)

A seguir veja a representação do conteúdo das tabelas na 3FN, com base nos dados originalmente expressos no relatório de alocação de docentes a projetos:

PROJETO

CODIGOPROJETO	TIPO	DESCRICAO
PRODATA	ANÁLISE DE DADOS	DESENVOLVIMENTO DE AMBIENTE PARA ANÁLISE
PROMED	ANÁLISE CLÍNICA	ATENDIMENTO COMUNITÁRIO E VACINAÇÃO

PROJETODOCENTE

CODIGOPROJETO	CODIGODOCENTE	DATAINICIO	TEMPOMESES
PRODATA	DOC001	01/02/2019	16

PRODATA	DOC002	01/02/2020	4
PRODATA	DOC003	01/02/2019	16
PRODATA	DOC004	01/02/2020	4
PROMED	DOC001	01/02/2019	16
PROMED	DOC010	01/06/2020	0
PROMED	DOC004	01/05/2020	1

DOCENTE

CODIGODOCENTE	NOME	CATEGORIA
DOC001	JOSÉ	ADJUNTO
DOC002	LUCIANO	TITULAR
DOC003	GILSON	ADJUNTO
DOC004	MARTA	TITULAR
DOC010	MARIA	ADJUNTO

CATEGORIA

CATEGORIA	SALARIO
ADJUNTO	R$ 6.000,00
TITULAR	R$ 16.000,00

Representação dos dados do relatório em tabelas na 3FN
Nathielly de Souza Campos

O modelo está na 3FN, pois, além de estar na 2FN, não existem dependências transitivas.

Neste módulo, nós aprendemos que o processo de normalização é útil para refinarmos a construção de um banco de dados relacional. Estudamos três formas normais e percebemos que, ao normalizarmos o nosso modelo, o número de tabelas tende a aumentar, minimizando a redundância nos dados.

Atividade 4

O processo de normalização objetiva eliminar dados redundantes, além de garantir que a dependência de dados faça sentido. A respeito da terceira forma normal em uma tabela relacional, assinale a alternativa verdadeira:

A

A tabela não precisa estar na 1FN.

B

A tabela não precisa estar na 2FN.

C

A tabela não pode conter campos opcionais

D

A tabela precisa estar na 2FN e não deve possuir coluna(s) com dependência(s) transitiva(s).

E

A tabela não pode possuir dependência transitiva de parte da chave primária.

A alternativa D está correta.

De fato, para estar na 3FN a tabela tem que respeitar as regras da 2FN, pois uma forma engloba as restrições da forma anterior. Além disso, não pode haver dependências entre colunas não-chave.

Normalização na prática

Agora, vamos realizar a normalização das tabelas de um sistema de controle de vendas que possui um arquivo de notas fiscais com a seguinte estrutura:

Arquivo de Notas Fiscais (Num. NF, Série, Data emissão, Cod. do Cliente, Nome do cliente, Endereço do cliente, CNPJ do cliente), Relação das mercadorias vendidas (em que, para cada mercadoria, temos: Código da Mercadoria, Descrição da Mercadoria, Quantidade vendida, Preço de venda e Total da venda desta mercadoria) e Total Geral da Nota.

Nesta apresentação, você vai praticar as formas normais 1FN, 2FN e 3FN.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Roteiro de prática

O processo de normalização ocorre, em geral, após a execução da modelagem lógica para validar a correção da modelagem.

Processo para obtenção da Primeira Forma Normal (1FN)

O processo de obtenção da Primeira Forma Normal objetiva eliminar grupos repetitivos e atributos compostos. Para isso, devemos:

- Definir chaves candidatas e escolher a chave primária da tabela.
- Transformar atributos compostos em atômicos.
- Transformar itens multivalorados em outras tabelas.

Processo para obtenção da Segunda Forma Normal (2FN)

O processo de obtenção da Segunda Forma Normal consiste em retirar das estruturas com chaves compostas (campo chave sendo formado por mais de um campo), os elementos funcionalmente dependentes de parte da chave. Para tal, devemos:

- Definir as colunas que não participaram da chave primária da tabela.

- Para cada uma das colunas identificadas, analisar se seu valor é determinado por parte ou pela totalidade da chave.

- Para as colunas dependentes parcialmente da chave: Criar tabelas nas quais a chave primária será a coluna da chave original que determinou o valor da coluna. Excluir da tabela original as colunas dependentes parcialmente da chave.
    - Criar tabelas nas quais a chave primária será a coluna da chave original que determinou o valor da coluna.
    - Excluir da tabela original as colunas dependentes parcialmente da chave.

Processo para obtenção da Terceira Forma Normal (3FN)

Consiste em retirar das estruturas os campos funcionalmente dependentes de outros campos que não são chaves. Para tal, devemos:

- Identificar as colunas que não participam da chave primária da tabela.

- Para cada uma das colunas identificadas, analisar se seu valor é determinado por alguma outra coluna não pertencente à chave.

- Para as colunas dependentes transitivamente da chave: Criar tabelas nas quais a chave primária será a coluna que determinou o valor da coluna analisada. Agregar a essas tabelas as colunas dependentes transitivamente. Excluir da tabela original as colunas dependentes transitivamente.
    - Criar tabelas nas quais a chave primária será a coluna que determinou o valor da coluna analisada.
    - Agregar a essas tabelas as colunas dependentes transitivamente.
    - Excluir da tabela original as colunas dependentes transitivamente.

Atividade 5

A imagem a seguir nos mostra uma situação de normalização em que as tabelas estão na Primeira Forma Normal e estão sendo analisadas quanto à Segunda Forma.

Projetos

Cod_P	Tipo	Descricao
LSC001	Novo Des.	Sistema de Estoque
PAG002	Manutenção	Sistema de RH

Proj_Emp

Cod_P	Cod_E	Nome	Categoria	Salario	Data_Inicio	Tempo_Alocacao
LSC001	2146	João	A1	4	01/11/2001	24
LSC001	3145	Silvio	A2	4	01/01/2002	24
LSC001	6126	José	B1	9	30/01/2002	12
LSC001	1214	Carlos	A2	4	15/05/2002	18
LSC001	8191	Mário	A1	4	10/02/2002	12
PAG002	8191	Mário	A1	4	05/10/2002	12
PAG002	4112	João M.	A1	4	10/12/2001	24
PAG002	6126	José	B1	9	30/01/2002	12

Projeto e empresas

Qual delas não precisará sofrer modificação para ficar na 2FN? Justifique sua resposta.

Chave de resposta

A tabela Projetos, como já está na 1FN e por possuir chave primária simples, está também automaticamente na Segunda Forma Normal. Isso porque não há como uma outra coluna depender apenas de parte da chave, como acontece em PROJ_EMP cujos dados de Empregado (Nome, Categoria e Salario) dependem apenas da coluna COD_E.

Mapeamento conceitual-lógico de entidades

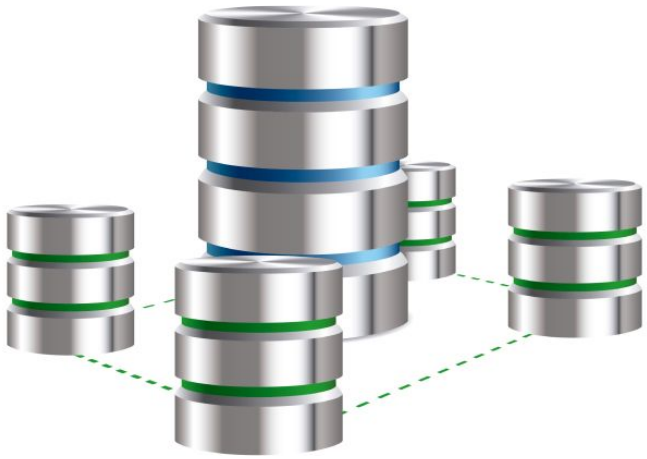
O mapeamento conceitual-lógico de entidades é uma etapa na modelagem de dados, servindo como ponte entre a abstração do modelo conceitual e a implementação concreta no modelo lógico. Com regras e técnicas específicas, as entidades e seus relacionamentos do modelo conceitual são transformados em estruturas de dados relacionais, viabilizando a manipulação e o armazenamento de informações de forma eficiente.

Neste vídeo, você vai compreender como realizar o mapeamento de relacionamentos no modelo lógico de um banco de dados relacional a partir de um DER.

> **Conteúdo interativo**
>
> Acesse a versão digital para assistir ao vídeo.

No projeto de banco de dados, há etapas que devem ser seguidas para a construção de um modelo conceitual. Para tanto, é usual a adoção do diagrama de entidade e relacionamento (DER) - tipo de modelo conceitual em que os principais objetos do negócio modelado se tornam entidades caracterizadas por atributos e relacionamentos.

Estudamos os principais componentes do modelo relacional, um modelo lógico que é base para a implementação de bancos de dados relacionais.



Rede de banco de dados vetorial

Neste módulo, construiremos a ponte entre os modelos conceitual e lógico.

Passaremos, então, a aplicar regras formais de mapeamento do modelo conceitual, visando a construção do modelo lógico. Na prática, projetaremos um banco de dados relacional a partir de um DER.

Regras de mapeamento

Podemos dividir o mapeamento conceitual-lógico em quatro etapas:

- Entidades
- Relacionamentos
- Atributos multivalorados
- Especialização/generalização

A seguir, conheceremos as regras de cada etapa do mapeamento e a sua respectiva aplicação por meio de exemplos.

Mapeamento de entidades

O mapeamento de entidades envolve:

Entidade forte ou independente

- Cada entidade E vira uma tabela T;
- Cada atributo simples da entidade E vira uma coluna na tabela T;
- O atributo identificador da entidade E vira chave primária na tabela T.

Entidade fraca ou dependente

- Cada entidade fraca F vira uma tabela T;
- Cada atributo simples da entidade F vira uma coluna na tabela T;
- A tabela T possuirá chave estrangeira originada a partir da chave primária da tabela proprietária;
- A tabela T possuirá chave primária composta pela coluna chave estrangeira criada no passo anterior e pela coluna mapeada do atributo identificador da entidade F na tabela T.

Exemplo de mapeamento de entidades

Conhecidas as regras para o mapeamento de entidades, observe no exemplo a seguir um diagrama de entidade e relacionamento (DER) contendo duas entidades:

FUNCIONARIO (entidade forte) e FONEFUNC (entidade fraca).

Diagrama de Entidade e Relacionamento (DER) contendo duas entidades.

A tabela a seguir exibe o modelo lógico gerado:

FUNCIONARIO		
CODIGOFUNCIONARIO	int	PK
NOME	char(90)	

FONEFUNC			
CODIGOFUNCIONARIO	int	PK	FK
NUMERO	char(15)	PK	
TIPO	char(15)	PK	

Criadas com base no mapeamento conceitual-lógico envolvendo entidade

Note que a tabela FUNCIONARIO corresponde à aplicação da regra para entidade forte. De modo semelhante, a tabela FONEFUNC corresponde à aplicação da regra para entidade fraca.

Após a aplicação das regras de mapeamento, foi gerada a representação textual a seguir:

FUNCIONARIO (CODGOFUNCIONARIO, NOME)
FONEFUNC (CODIGOFUNCIONARIO, NUMERO, TIPO)
CODIGOFUNCIONARIO REFERENCIA FUNCIONARIO

Atividade 1

No mapeamento conceitual-lógico de entidades, qual das alternativas inclui a etapa que ocorre após a transformação do modelo conceitual em um modelo lógico relacional?

A

Identificação das entidades e seus atributos.

B

Definição das cardinalidades dos relacionamentos entre as entidades.

C

Conversão das entidades em tabelas e dos atributos em colunas.

D

Normalização das tabelas para eliminar redundâncias e anomalias.

E

Implementação do modelo lógico em um sistema de gerenciamento de banco de dados (SGBD).

A alternativa E está correta.

O mapeamento conceitual-lógico de entidades consiste em quatro etapas principais: identificação das entidades e seus atributos; definição das cardinalidades dos relacionamentos entre as entidades; conversão das entidades em tabelas e dos atributos em colunas; normalização das tabelas para eliminar redundâncias e anomalias.

A implementação do modelo lógico em um SGBD é a etapa final que ocorre após o mapeamento conceitual-lógico.

Mapeamento de relacionamentos

O mapeamento conceitual-lógico de relacionamentos envolve transformar as interconexões entre entidades do modelo conceitual em estruturas de dados relacionais. Por meio de técnicas precisas, os diversos tipos de relações, como um para um, um para muitos e muitos para muitos, são mapeados para cardinalidades e chaves estrangeiras no modelo lógico, garantindo a precisão e a consistência da informação.

Neste vídeo, você vai compreender como realizar o mapeamento de relacionamentos no modelo lógico de um banco de dados relacional a partir de um DER.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O mapeamento de relacionamentos dependerá da cardinalidade máxima:

1

Relacionamentos 1:1

- Cardinalidade (0,1):(0,1): priorizar adição de coluna(s). Alternativa: tabela própria.
- Cardinalidade (0,1):(1,1): priorizar fusão de tabelas. Alternativa: adição de colunas.
- Cardinalidade (1,1):(1,1): fusão de tabelas.

2

Relacionamentos 1:n

- Identificar a tabela T do lado N.
- Adicionar chave estrangeira na tabela T do lado N referente à chave primária da tabela do lado 1.
- Cada atributo simples do relacionamento vira uma coluna na tabela T.

3

Relacionamentos n:n

- Cada relacionamento vira uma tabela T.
- A tabela T possuirá chaves estrangeiras originadas das chaves primárias das tabelas participantes do relacionamento.
- A tabela T possuirá chave primária composta pelas chaves estrangeiras criadas no passo anterior.
- Cada atributo simples do relacionamento vira uma coluna na tabela T.

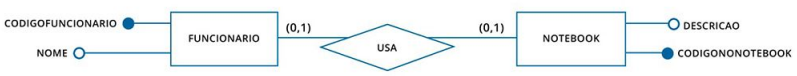
4

Relacionamentos n-ários (análogo a relacionamentos n:n)

- Cada relacionamento vira uma tabela T.
- A tabela T possuirá chaves estrangeiras originadas das chaves primárias das tabelas participantes do relacionamento.
- A tabela T possuirá chave primária composta pelas chaves estrangeiras criadas no passo anterior.
- Cada atributo simples do relacionamento vira uma coluna na tabela T.

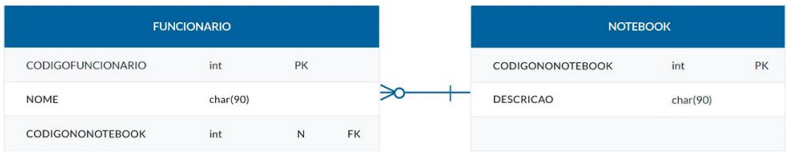
Exemplos de mapeamento de relacionamento 1:1

Conhecidas as regras para o mapeamento de relacionamentos 1:1, observe no exemplo a seguir um diagrama de entidade e relacionamento (DER) contendo duas entidades: FUNCIONARIO e NOTEBOOK. Há um relacionamento 1:1 no qual ambas as entidades possuem participação **opcional** (cardinalidades (0,1): (0,1)).



DER contendo relacionamento 1:1 – ambas as entidades com participação opcional

A tabela a seguir exibe o modelo lógico gerado:



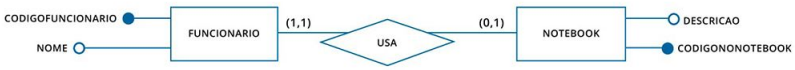
Criadas com base no mapeamento conceitual-lógico envolvendo relacionamento 1:1 - ambas as entidades com participação opcional

Para esse tipo de relacionamento, o mais adequado é priorizar adição de coluna(s), o que ocorreu ao criarmos a coluna CODIGONOTEBOOK como chave estrangeira na tabela FUNCIONARIO. Vale lembrar que estamos diante de uma coluna opcional.

A seguir, a representação textual do modelo:

NOTEBOOK (CODIGONOTEBOOK, DESCRICAO)
FUNCIONARIO (CODIGOFUNCIONARIO, NOME, CODIGONOTEBOOK)
CODIGONOTEBOOK REFERENCIA NOTEBOOK

A imagem a seguir apresenta um DER contendo um relacionamento 1:1 no qual há uma entidade com participação obrigatória e a outra opcional (cardinalidades (0,1): (1,1)).



DER contendo relacionamento 1:1 – uma entidade com participação obrigatória e a outra opcional

A tabela a seguir exibe o modelo lógico gerado:

FUNCIONARIO		
CODIGOFUNCIONARIO	int	PK
NOME	char(90)	
CODIGONONOTEBOOK	int	N
DESCRICAO	char(90)	N

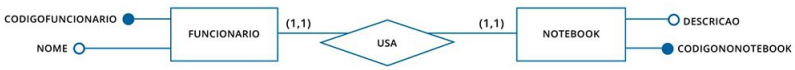
Criada com base no mapeamento conceitual-lógico envolvendo relacionamento 1:1 - uma entidade com participação obrigatória e a outra opcional
Nathielly de Souza Campos

Para esse tipo de relacionamento, o mais adequado é priorizar fusão de tabelas, o que ocorreu ao criarmos as colunas CODIGONOTEBOOK e DESCRICAO na tabela FUNCIONARIO, ambas opcionais.

Após a aplicação das regras de mapeamento, foi gerada a representação textual a seguir:

FUNCIONARIO (CODIGOFUNCIONARIO, NOME, CODIGONOTEBOOK, DESCRICAO)

A imagem a seguir apresenta um DER parcial contendo um relacionamento 1:1 e ambas as entidades com participação obrigatória (cardinalidades (1,1): (1,1)).



DER parcial contendo relacionamento 1:1 – ambas as entidades com participação obrigatória

A tabela a seguir exibe o modelo lógico gerado:

FUNCIONARIO		
CODIGOFUNCIONARIO	int	PK
NOME	char(90)	
CODIGONONOTEBOOK	int	
DESCRICAO	char(90)	

Criadas com base no mapeamento conceitual-lógico envolvendo relacionamento 1:1 – ambas as entidades com participação obrigatória
Nathielly de Souza Campos

Para esse tipo de relacionamento, o mais adequado é priorizar fusão de tabelas, o que ocorreu ao criarmos as colunas CODIGONOTEBOOK e DESCRICAO na tabela FUNCIONARIO, ambas obrigatórias.

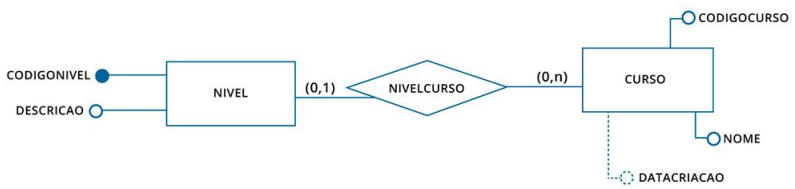
Após a aplicação das regras de mapeamento, foi gerada a representação textual a seguir:

FUNCIONARIO (CODIGOFUNCIONARIO, NOME, CODIGONOTEBOOK, DESCRICAO)

Exemplo de mapeamento de relacionamento 1:N

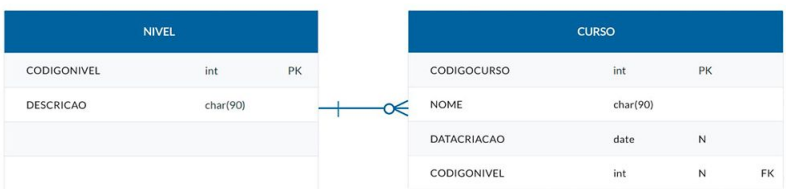
Conhecidas as regras para o mapeamento de relacionamentos 1:N, observe no exemplo a seguir um diagrama de entidade e relacionamento (DER) contendo duas entidades: NIVEL e CURSO.

A imagem a seguir mostra um DER parcial contendo um relacionamento **1:N**.



DER contendo relacionamento 1:N

A imagem a seguir exibe o modelo lógico gerado:



Criadas com base no mapeamento conceitual-lógico envolvendo relacionamento 1:N

Para esse tipo de relacionamento, foi utilizada adição de coluna(s) na tabela do lado N, o que ocorreu ao criarmos a chave estrangeira CODIGONIVEL na tabela CURSO.

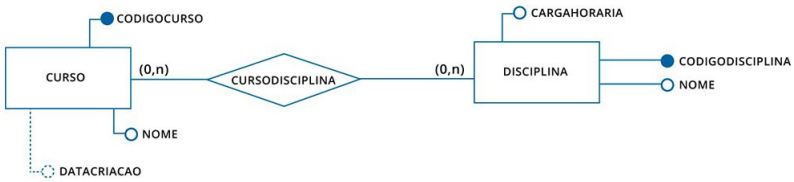
Após a aplicação das regras de mapeamento, foi gerada a representação textual a seguir:

```
NIVEL (CODGONIVEL, DESCRICAO)
 CURSO (CODIGOCURSO, NOME, DATACRIACAO, CODIGONIVEL)
  CODIGONIVEL REFERENCIA NIVEL
```

Exemplo de mapeamento de relacionamento N:N

Conhecidas as regras para o mapeamento de relacionamentos N:N, observe no exemplo a seguir um diagrama de entidade e relacionamento (DER) contendo duas entidades: CURSO e DISCIPLINA.

A imagem a seguir mostra um DER parcial contendo um relacionamento **N:N**.



DER parcial contendo relacionamento N:N

A imagem a seguir exibe o modelo lógico gerado:

Criadas com base no mapeamento conceitual-lógico envolvendo relacionamento N:N

Para esse tipo de relacionamento, foi utilizada tabela própria, o que ocorreu ao criarmos CURSODISCIPLINA, contendo duas chaves estrangeiras: CODIGOCURSO e CODIGODISCIPLINA. Ao mesmo tempo, a combinação das duas colunas forma a chave primária composta da tabela.

Após a aplicação das regras de mapeamento, foi gerada a representação textual a seguir:

```
CURSO (CODIGOCURSO, NOME, DATACRIACAO, CODIGONIVEL)
 CODIGONIVEL REFERENCIA NIVEL
  DISCIPLINA (CODIGODISCIPLINA, NOME, CARGAGORARIA)
    CURSODISCIPLINA (CODIGOCURSO, CODIGODISCIPLINA)
      CODIGOCURSO REFERENCIA CURSO
        CODIGODISCIPLINA REFERENCIA DISCIPLINA
```

A observação sobre entidade associativa e autorrelacionamento no campo da modelagem de bancos de dados é fundamental para entender a complexidade dos relacionamentos entre diferentes conjuntos de dados. Vejamos cada um deles:

Observação sobre entidade associativa

O mapeamento de entidade associativa, isto é, relacionamento com atributos, segue as regras utilizadas no mapeamento de relacionamentos N:N.

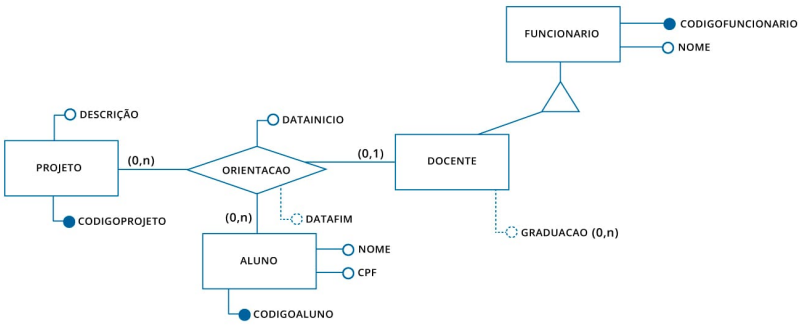
Observação sobre autorrelacionamento

O mapeamento de autorrelacionamento segue as regras utilizadas no mapeamento de relacionamentos. Basta, então, você ficar atento ao tipo de cardinalidade máxima em questão.

Exemplo de mapeamento de relacionamento ternário

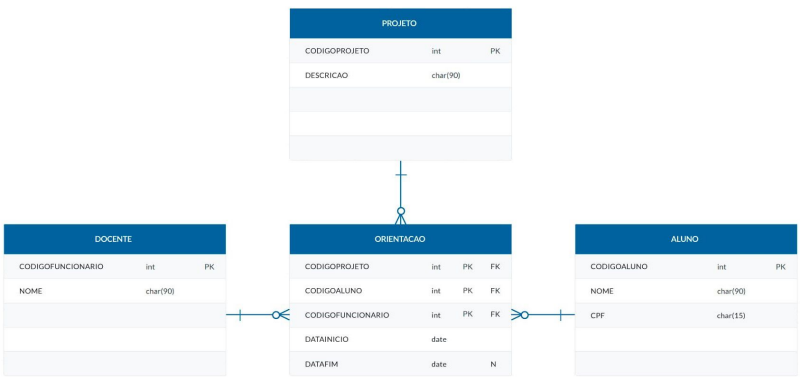
Conhecidas as regras para o mapeamento de relacionamentos, ao nos depararmos com relacionamentos ternários, precisaremos avaliar as cardinalidades máximas em questão. Observe no exemplo a seguir um diagrama de entidade e relacionamento (DER) contendo um relacionamento ternário entre as entidades PROJETO, DOCENTE e ALUNO.

A imagem a seguir mostra um DER parcial contendo um relacionamento **ternário**.



DER parcial contendo relacionamento ternário

A tabela a seguir exibe o modelo lógico gerado:



Tabelas criadas com base no mapeamento conceitual-lógico envolvendo relacionamento ternário

Para esse tipo de relacionamento, foi utilizada tabela própria, o que ocorreu ao criarmos ORIENTACAO, contendo três chaves estrangeiras: CODIGOFUNCIONARIO, CODIGOALUNO e CODIGOPROJETO. Além disso, foram criadas as colunas DATAINICIO e DATAFIM, as quais representam informações importantes sob o contexto de um registro de orientação.

Após a aplicação das regras de mapeamento, foi gerada a representação textual a seguir:

PROJETO (CODIGOPROJETO, DESCRICAO)
DOCENTE (CODIGOFUNCIONARIO, NOME)

```
ALUNO (CODIGOALUNO, NOME, CPF)
  ORIENTACAO (CODIGOPROJETO, CODIGOALUNO, CODIGOFUNCIONARIO, DATAINICIO, DATAFIM)
    CODIGOPROJETO REFERENCIA PROJETO
      CODIGOALUNO REFERENCIA ALUNO
        CODIGOFUNCIONARIO REFERENCIA DOCENTE
```

Atividade 2

Considere um relacionamento entre TURMA e DISCIPLINA (N:N) denominado OFERTA. Para cada disciplina ofertada por uma turma é necessário saber o número de vagas. Qual alternativa a seguir representa corretamente o modelo lógico derivado do modelo conceitual

A

Somente uma tabela para todo o modelo.

B

Duas tabelas, uma para TURMA e outra para DISCIPLINA.

C

Três tabelas: uma para TURMA, uma para DISCIPLINA e uma para OFERTA. Além disso, o atributo que representa o número de vagas deve estar modelado na tabela OFERTA.

D

Três tabelas: uma para TURMA, uma para DISCIPLINA e uma para OFERTA. Além disso, o atributo que representa o número de vagas deve estar modelado na tabela DISCIPLINA.

E

Três tabelas: uma para TURMA, uma para DISCIPLINA e uma para OFERTA. Além disso, o atributo que representa o número de vagas deve estar modelado na tabela TURMA.

A alternativa C está correta.

Diante da cardinalidade máxima N:N, a solução é criar três tabelas: duas para as entidades participantes e uma para o relacionamento. O atributo que representa o número de vagas, pelo fato de fazer parte do relacionamento, deve ser modelado na tabela OFERTA.

Mapeamento de atributos multivalorados

O mapeamento de atributos multivalorados é uma técnica para lidar com dados complexos em bancos de dados relacionais. Com métodos específicos, os atributos que podem assumir múltiplos valores para uma mesma entidade são mapeados para estruturas de dados eficientes, como tabelas relacionadas ou arrays, garantindo a flexibilidade e a eficiência na manipulação da informação.

Neste vídeo, você vai compreender como realizar o mapeamento de atributos multivalorados no modelo lógico de um banco de dados relacional a partir de um DER.

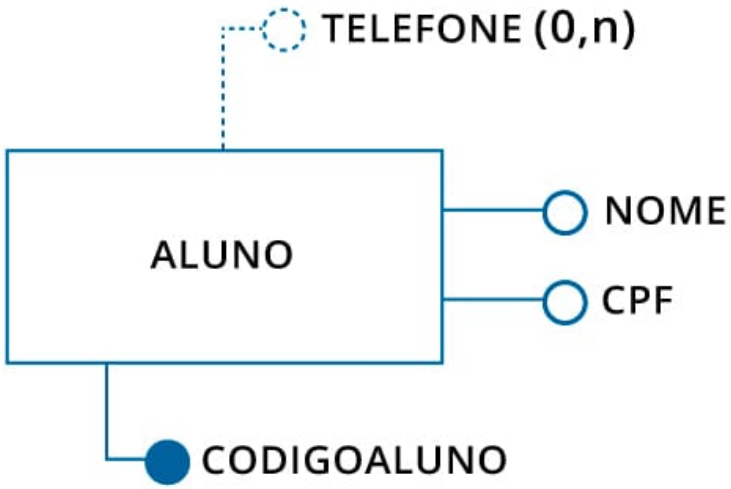
Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O mapeamento de atributos multivalorados envolve:

- Criar uma tabela T para cada atributo multivalorado.
- Criar coluna(s) para o(s) atributo(s) multivalorado(s).
- A tabela T possuirá chave estrangeira originada da chave primária da tabela original.
- A tabela T possuirá chave primária composta pela chave estrangeira criada no passo anterior e pela(s) coluna(s) referente(s) ao(s) atributo multivalorado(s).

A imagem a seguir mostra um DER parcial contendo atributo multivalorado.



DER parcial contendo atributo multivalorado

A tabela a seguir exibe o modelo lógico gerado:

ALUNO		
CODIGOALUNO	int	PK
NOME	char(90)	
CPF	char(15)	

FONEALUNO			
CODIGOALUNO	int	PK	FK
NUMERO	char(15)	PK	
TIPO	char(15)	PK	

Criadas com base no mapeamento conceitual-lógico envolvendo atributo multivalorado

Note que para mapear o atributo multivalorado TELEFONE, foi criada tabela própria denominada FONEALUNO. A coluna CODIGOALUNO de FONEALUNO é chave estrangeira. Além disso, as colunas CODIGOALUNO, NUMERO, TIPO representam uma chave primária composta. A coluna TIPO, criada na tabela FONEALUNO, representa a categoria do telefone, por exemplo, residencial, comercial ou móvel.

Após a aplicação das regras de mapeamento, foi gerada a representação textual a seguir:

```
ALUNO (CODIGOALUNO, NOME, CPF)
  FONEALUNO (CODIGOALUNO, NUMETO, TIPO)
    CODIGOALUNO REFERENCIA ALUNO
```

Atividade 3

No mapeamento de atributos multivalorados, avalie as técnicas utilizadas para lidar com a multiplicidade de valores em um banco de dados relacional a seguir:

I. Criação de uma nova tabela para armazenar os valores multivalorados.

II. Armazenamento dos valores multivalorados em um único campo, separados por um delimitador.

III. Normalização da tabela para evitar redundância e anomalias de dados.

IV. Criação de uma coluna separada para cada valor multivalorado.

Está correto o que se afirma em:

A

I, II, III e IV.

B

II, III e IV.

C

I, III e IV.

D

I, II e IV.

E

I, II e III.

A alternativa D está correta.

O mapeamento de atributos multivalorados lida com a multiplicidade de valores de forma eficiente, utilizando as seguintes técnicas: criação de uma nova tabela; armazenamento em um único campo; colunas separadas.

A normalização, por outro lado, é uma técnica para evitar redundâncias e anomalias de dados, e não se aplica diretamente ao mapeamento de atributos multivalorados.

Mapeamento de especialização/generalização

O mapeamento de especialização/generalização é uma técnica avançada de modelagem de dados que permite representar hierarquias complexas entre entidades. Por meio de regras e métodos específicos, as

entidades genéricas e suas subespecializações são mapeadas para estruturas de dados relacionais, evidenciando as nuances e os relacionamentos de herança entre classes de objetos.

Neste vídeo, você vai ver como realizar o mapeamento da estrutura de especialização/generalização no modelo lógico de um banco de dados relacional a partir de um DER.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O mapeamento de especialização/generalização envolve:

Solução I

Tabela única.

- Criar uma tabela única que contenha todos os atributos das entidades genérica e especializadas.
- Criar uma coluna TIPO, caso não exista, para identificar a entidade especializada.

Solução II

Uma tabela para cada entidade (genérica ou especializada) que compõe a hierarquia.

- Criar uma tabela para a entidade genérica, com coluna(s) referente(s) ao(s) atributo(s) da entidade genérica.
- Criar uma tabela para cada entidade especializada, com coluna(s) referentes ao(s) atributo(s) da entidade especializada.
- Cada tabela de entidade especializada possuirá chave estrangeira originada da chave primária da tabela da entidade genérica. Esta também será a sua chave primária.

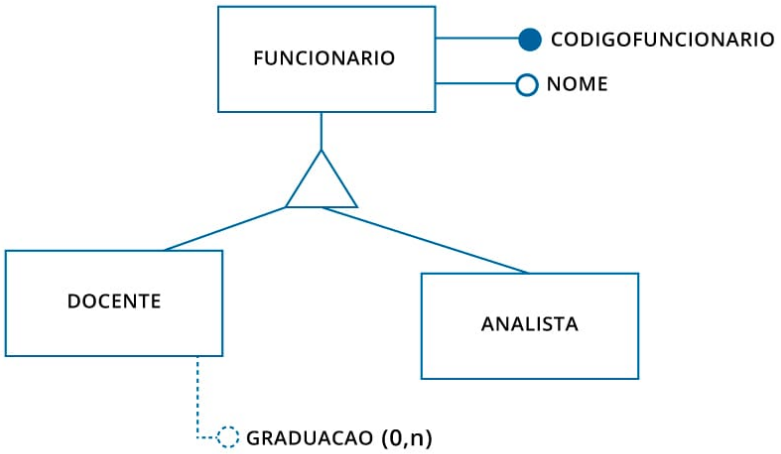
Solução III

Subdivisão da entidade genérica.

- Nesta implementação, não será criada tabela para a entidade genérica.
- Criar uma tabela para cada entidade especializada, com coluna(s) referentes ao(s) atributo(s) da entidade especializada.
- Em cada tabela, criar colunas referentes aos atributos da entidade genérica, sendo sua chave primária originada do atributo identificador da entidade genérica.
- Caso a hierarquia seja parcial, será necessário criar uma tabela adicional para abarcar as entidades genéricas que não estão nas entidades especializadas. Essa tabela conterá somente os atributos da entidade genérica.
- Importante notar que esta solução pode gerar redundância de dados, no caso de hierarquia sobreposta, requerendo um tratamento adicional para controle de redundância.

Exemplos de mapeamento de especialização/generalização

Conhecidas as regras para o mapeamento de **especialização/generalização**, observe no exemplo a seguir um diagrama de entidade e relacionamento (DER) com esse mecanismo:



DER contendo especialização/generalização

A imagem a seguir exibe o modelo lógico gerado após a aplicação da **solução I** expressa nas regras de mapeamento:

FUNCIONARIO		
CODIGOFUNCIONARIO	int	PK
NOME	char(90)	
TIPO	char(40)	

Criada com base no mapeamento conceitual-lógico envolvendo especialização/generalização – solução I
Nathielly de Souza Campos

No exemplo, foi criada tabela única (FUNCIONARIO) contendo colunas referentes à entidade genérica, além da coluna TIPO, para identificar a categoria de cada colaborador.

Após a aplicação das regras de mapeamento, foi gerada a representação textual a seguir:

FUNCIONARIO (CODIGOFUNCIONARIO, NOME, TIPO)

A imagem a seguir exibe o modelo lógico gerado após a aplicação da solução II das regras de mapeamento:

Criadas com base no mapeamento conceitual-lógico envolvendo especialização/generalização – solução II

No exemplo, foram criadas três tabelas: uma (FUNCIONARIO) referente à entidade genérica. As restantes, DOCENTE e ANALISTA, referentes às entidades especializadas em questão. Cada CODIGOFUNCIONARIO presente nas tabelas DOCENTE e ANALISTA exerce o papel de chave estrangeira.

Após a aplicação das regras de mapeamento, foi gerada a representação textual a seguir:

FUNCIONARIO (CODIGOFUNCIONARIO, NOME)
DOCENTE (CODIGOFUNCIONARIO)
CODIGOFUNCIONARIO REFERENCIA FUNCIONARIO
ANALISTA (CODIGOFUNCIONARIO)
CODIGOFUNCIONARIO REFERENCIA FUNCIONARIO

A tabela a seguir exibe o modelo lógico gerado após a aplicação da solução III das regras de mapeamento:

Tabelas criadas com base no mapeamento conceitual-lógico envolvendo especialização/generalização – solução III.

No exemplo, foram criadas três tabelas, sendo OUTROSFUNCIONARIOS onde devem ser mantidos funcionários que não sejam docentes nem analistas. As restantes, DOCENTE e ANALISTA, são referentes às entidades especializadas em questão.

Após a aplicação das regras de mapeamento, foi gerada a representação textual a seguir:

DOCENTE (CODIGOFUNCIONARIO, NOME)
ANALISTA (CODIGOFUNCIONARIO, NOME)
OUTROSFUNCIONARIOS (CODIGOFUNCIONARIO, NOME)

Convém ressaltar que a tabela OUTROSFUNCIONARIOS seria necessária somente se o mecanismo de generalização/especialização fosse parcial, ou seja, se houvesse algum colaborador não enquadrado nas categorias docente ou analista.

Atenção

Como dica prática, devemos ter cuidado especial caso seja escolhida a solução III. Ao incluir um novo funcionário, será necessário verificar todas as tabelas criadas para as especializações para garantir a unicidade da chave primária. No exemplo, é preciso verificar os valores de chave primária nas tabelas DOCENTE, ANALISTA e OUTROSFUNCIONARIOS.

Convém ressaltar que a solução II é a mais usual, por ser mais flexível, dada a facilidade existente em contemplar novas especializações. Além disso, a solução I tende a gerar diversas ocorrências de valores nulos em colunas. Ao mesmo tempo, a solução III apresenta maior possibilidade de gerar redundância de dados.

Atividade 4

Questão 1

Considere um modelo conceitual de banco de dados que representa a relação entre funcionários e seus cargos. A entidade Funcionário tem os atributos Nome, CPF e Salário. A entidade Cargo apresenta os atributos NomeCargo e Departamento. As entidades Gerente e Vendedor são especializações da entidade Funcionário. A entidade Gerente possui o atributo Bônus, enquanto a entidade Vendedor, o atributo Comissão.

Avalie as implementações do mapeamento conceitual/lógico de especialização/generalização, com herança para essa situação, propostas a seguir:

I. Criar uma única tabela Funcionario com todos os atributos das entidades Funcionario, Gerente e Vendedor.

II. Criar uma tabela Funcionario com os atributos Nome, CPF e Salário, e uma tabela Cargo com os atributos NomeCargo e Departamento.

III. Criar uma tabela Gerente com os atributos Nome, CPF, Salário, NomeCargo, Departamento e Bônus.

IV. Criar uma tabela Funcionario com os atributos Nome, CPF e Salário, e uma tabela Cargo com os atributos NomeCargo, Departamento, Bônus e Comissão.

Está correto o que se afirma em:

A

I, II, III e IV.

B

II, III e IV.

C

I, III e IV.

D

I, II e IV.

E

I, II e III.

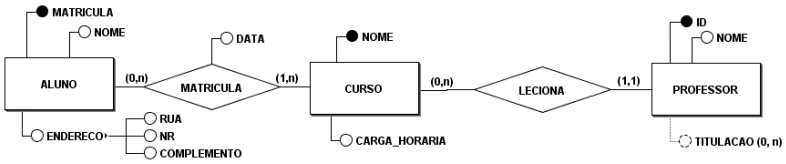
A alternativa E está correta.

A implementação correta do mapeamento conceitual/lógico de especialização/generalização com herança para essa situação apresenta-se da seguinte forma.

- Tabelas separadas: Uma tabela Funcionario é criada com os atributos Nome, CPF e Salário, e uma tabela Cargo é criada com os atributos NomeCargo e Departamento.
- Herança: As entidades Gerente e Vendedor herdam os atributos Nome, CPF e Salário da entidade Funcionario.
- Atributos específicos: A tabela Gerente possui o atributo Bônus, enquanto a tabela Vendedor possui o atributo Comissão.

Estudo de caso de mapeamento conceitual-lógico

Agora, vamos executar a modelagem lógica da Empresa de Treinamento cuja modelagem conceitual está na imagem a seguir.



Representação da modelagem lógica da Empresa de Treinamento

Se desejar, você pode realizar o download do arquivo do BRMODELO3.0: EMPTREINCONCEITUAL.BRM3.

Estudo de caso de mapeamento conceitual-lógico

Neste vídeo, você vai praticar o mapeamento conceitual-lógico a partir de um DER.

> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

Roteiro de prática

Para realizarmos a modelagem lógica para o modelo relacional e a modelagem conceitual representada em um DER, devemos acompanhar os seguintes passos.

- Passo 1: Para cada entidade tipo, criar uma tabela.
- Passo 2: Para cada atributo simples e monovalorado, gerar uma coluna na tabela de entidade tipo.
- Passo 3: Marcar atributos únicos e obrigatórios (identificadores) nas tabelas criadas.
- Passo 4: Marcar as chaves primárias das tabelas criadas.
- Passo 5: Tratar atributos compostos decompondo-os como colunas na tabela que se originou de sua entidade tipo.
- Passo 6: Tratar atributos multivalorados criando uma nova tabela com chave estrangeira para tabela original.
- Passo 7: Tratar relacionamentos N:N criando uma tabela que recebe as chaves das tabelas envolvidas no relacionamento.
- Passo 8: Tratar relacionamentos 1:1 e 1:N levando a chave estrangeira do lado 1 para o lado N ou para o outro lado no caso de 1:1.
- Passo 9: Definir o tipo das colunas.
- Passo 10: Tratar a generalização/especialização.

Atividade 5

Responda às perguntas a seguir:

a) Quando vamos determinar a chave primária de uma tabela, que tipo de atributos devemos procurar como chave candidata?

b) Se existir mais de uma chave candidata, como devemos agir?

c) Se não existir chave candidata na tabela, como devemos proceder?

Chave de resposta

a) Uma chave primária deve ser um atributo único e de preenchimento obrigatório; dessa forma, devemos procurar atributos que tenham as características de serem único e não nulos, pois eles são chaves candidatas.

b) Se existir mais de uma chave candidata na tabela, devemos eleger uma como chave primária; as demais continuam como atributos único e não nulos. Devemos escolher, a princípio, aquela candidata que seja o identificador natural dentro do minimundo. Por exemplo, se a tabela ALUNO possui as chaves candidatas CPF e Matrícula, devemos eleger matricula, pois ela é o identificador natural do aluno.

c) Se não existe chave candidata, devemos criar uma coluna na tabela para exercer essa função. Esta coluna não, em atributo correspondente no modelo conceitual, é apenas uma exigência da implementação do modelo relacional.

Consultas e transações em banco de dados

Este núcleo faz uma introdução abrangente aos conceitos que sustentam o funcionamento de bancos de dados: consultas e transações. Vamos abordar cada um desses pilares em detalhes, explorando definições, características, benefícios e implicações para o desempenho e a confiabilidade do sistema.

Neste vídeo, você vai compreender os pilares do funcionamento de um banco de dados, ou seja, consultas e transações, incluindo definições, características, benefícios e implicações para o desempenho e a confiabilidade do sistema.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Consultas

A partir de agora, conheceremos diretrizes que devem ser consideradas quando formos implementar um banco de dados relacional. As diretrizes abrangem aspectos que influenciam no desempenho do banco de dados. Assim, o projeto lógico pode sofrer ajustes para adaptar-se ao sistema gerenciador de banco de dados (SGBD) escolhido para a implementação.

Homem trabalhando banco de dados relacional enquanto fala ao telefone

Planejar um banco de dados que tenha um bom desempenho pressupõe adquirir conhecimento sobre as consultas e transações que serão realizadas pela aplicação.

Um SGBD tipicamente processa e devolve dados requisitados em consultas para diversas finalidades, tais como recuperação, inclusão, exclusão ou mesmo atualização de dados. As consultas são implementadas com o auxílio da linguagem SQL (do Inglês, *Structured Query Language* – linguagem de consulta estruturada).

Dica

Um código típico de consulta para recuperar dados em SQL envolve o comando SELECT com as cláusulas FROM e WHERE.

Por exemplo, dada a tabela DOCENTE (CODIGODOCENTE, NOME, SEXO), o código a seguir recupera os registros de todas as professoras:

```
SELECT CODIGODOCENTE, NOME
  FROM DOCENTE
    WHERE SEXO='F
```

A primeira linha do comando informa ao SGBD as colunas que devem ser exibidas após o processamento da consulta. Na segunda, especificamos o nome da tabela que contém os dados. Finalmente, na terceira linha, adicionamos uma condição de filtro que será processada pelo SGBD para recuperar as linhas de interesse.

Transações

Diversos SGBDs modernos permitem a especificação de operações de transação.

> Uma transação corresponde a uma série de operações que, quando submetidas ao SGBD, devem ser consideradas como uma unidade lógica de trabalho. Isso significa que todas as operações que compõem uma transação precisam ser executadas. Caso contrário, é necessário serem canceladas e nenhuma modificação ocorrerá no banco de dados.

Por exemplo, em um processo de inscrição em disciplinas, em geral, o aluno tem a liberdade para compor seu quadro de horário de disciplinas, para, em seguida, confirmar inscrição em diversas matérias. Assim, a inscrição em disciplinas deve ser considerada como um único processo ou transação. Trata-se de um procedimento atômico: ou todas as operações são confirmadas ou nenhuma delas é realizada.

Atividade 1

Em um sistema bancário, avalie as características fundamentais das transações bancárias a seguir:

I. Reversibilidade: As transações podem ser revertidas para um estado anterior caso alguma operação falhe.

II. Atomicidade: A transação deve ser executada como um todo, ou seja, todas as suas operações devem ser concluídas com sucesso ou nenhuma delas será.

III. Consistência: A transação deve manter a consistência dos dados do banco de dados, mesmo em caso de falhas.

IV. Isolamento: As transações devem ser executadas como se estivessem sozinhas, sem interferir umas nas outras.

Está correto o que se afirma em:

A

I, II, III e IV.

B

II, III e IV.

C

I, III e IV.

D

I, II e IV.

E

I, II e III.

A alternativa B está correta.

As características fundamentais das transações bancárias são:

Atomicidade, Consistência , Isolamento e Durabilidade. A reversibilidade não é uma característica fundamental das transações. Em alguns casos, pode ser necessário reverter uma transação manualmente, mas isso não é um processo automático.

Indexação e consultas envolvendo mais de uma tabela

Aqui abordaremos os pilares que sustentam a recuperação eficiente de dados em bancos de dados relacionais: indexação e consultas com múltiplas tabelas. Vamos explorar como a indexação otimiza o acesso aos dados, por meio de diferentes tipos de índices e suas aplicações. Também vamos nos aprofundar na construção de consultas complexas que combinam dados de várias tabelas, utilizando joins e outras técnicas avançadas.

Neste vídeo, você vai conhecer os pilares que sustentam a recuperação eficiente de dados em bancos de dados relacionais, a indexação e as consultas com múltiplas tabelas.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Consultas envolvendo mais de uma tabela

Você perceberá que, em geral, a maior parte das consultas para recuperar informações de um banco de dados envolve diversas tabelas. Isso ocorre, principalmente, quando o projeto leva em conta as regras de normalização.

Considere a estrutura de duas tabelas relacionadas, conforme a seguir:

```
MUNICIPIO (CO_MUNICIPIO, NOME)
  DM_DOCENTE_2 (CO_DOCENTE_IES, CO_IES, NO_IES, CO_MUNICIPIO_NASCIMENTO)
    CO_MUNICIPIO_NASCIMENTO REFERENCIA MUNICIPIO
```

A relação entre as tabelas está representada pela coluna de chave estrangeira CO_MUNICIPIO_NASCIMENTO da tabela DM_DOCENTE_2, a qual faz referência para a coluna chave primária CO_MUNICIPIO da tabela MUNICIPIO.

Nosso objetivo é recuperar o código do docente e nome do município de nascimento dele.

Perceba que as colunas alvo do resultado estão presentes em tabelas distintas: NOME, na tabela MUNICIPIO e CO_DOCENTE_IES, na tabela DM_DOCENTE_2.

O código em SQL que recupera os dados de interesse está expresso a seguir:

```
SELECT CODIGO_DOCENTE_IES, NOME
  FROM DM_DOCENTE_2, MUNICIPIO
```

```
WHERE
  MUNICIPIO.CO_MUNICIPIO=DM_DOCENTE_2.CO_MUNICIPIO_NASCIMENTO;
```

A primeira linha do comando serve para declararmos as colunas que farão parte do resultado da consulta. Na segunda, informamos as tabelas de interesse. Finalmente, na última linha, há uma condição de filtro, envolvendo uma igualdade entre a chave primária da tabela MUNICIPIO e a chave estrangeira da tabela DM_DOCENTE_2.

Para processar a consulta anterior, o SGBD cria uma estrutura de tabela temporária que contém a combinação de cada linha da tabela DM_DOCENTE_2 com cada linha da tabela MUNICIPIO. Se considerarmos os 367.980 registros de DM_DOCENTE_2 e os 5.570 registros da tabela MUNICIPIO, a tabela temporária teria mais de dois bilhões de linhas (367.980*5.570).

Finalmente, a partir da tabela temporária, o SGBD executa o filtro especificado no comando WHERE para então exibir as colunas listadas no comando SELECT. O processo anterior é bastante custoso para o SGBD, ainda que cada sistema internamente use técnicas para otimizar o processamento.

Atenção

Note que, se esse tipo de consulta for frequente, haverá grande probabilidade de lentidão no sistema.

Indexação em banco de dados

O desempenho de consultas é um assunto vasto que faz uso de diversas estratégias de acesso a dados, semelhantes às utilizadas em nosso dia a dia.

Por exemplo, ao buscarmos por determinada informação em algum livro, para *ganharmos tempo*, é comum primeiro consultar o índice remissivo do livro, que indicará a página onde se localiza o termo buscado.

Atenção

Em banco de dados, índices funcionam como estruturas auxiliares utilizadas para tornar mais eficiente a recuperação de registros em resposta a determinadas condições de busca.

Normalmente, ao projetamos uma tabela com chave primária, os registros de dados são gravados em disco sem nenhum critério de ordenação das linhas da tabela. Para facilitar a consulta pelo valor da chave primária, o SGBD cria uma estrutura de índice para a chave primária de cada tabela.

A estrutura de índice poderá ser utilizada pelo SGBD caso seja necessário realizar consulta que envolva, por exemplo, uma condição de igualdade na coluna de chave primária da tabela. Quando isso ocorre, o desempenho da consulta em geral é melhor do que caso não existisse a estrutura de índice.

Exemplo prático envolvendo indexação

Visando ressaltar a importância dos índices, realizamos um pequeno experimento, que consiste em submeter duas consultas ao SGBD, uma sem índice, e a segunda com uma coluna indexada.

> Vamos perceber que, quando o SGBD processa uma consulta com o auxílio de um índice, o tempo de resposta tende a ser mais otimizado se comparado à execução da mesma consulta sem esse recurso.

Suponha então a existência de uma tabela DM_DOCENTE (CO_IES, NO_IES, CO_DOCENTE_IES, CO_MUNICIPIO_NASCIMENTO) - apresentada aqui com quatro colunas para fins de exemplo – originalmente, extraída do Censo da Educação Superior Brasileira de 2016.

A tabela contém 367.980 registros. Cada registro corresponde a um docente vinculado a uma instituição de ensino superior (IES). Ainda, originalmente, os registros de DM_DOCENTE estão fisicamente ordenados pela coluna CO_IES e a tabela não possui chave primária definida.

Nosso objetivo é recuperar todas as colunas da tabela, referentes ao docente que possui o valor 850516 para a coluna CO_DOCENTE_IES.

O comando SQL executado na consulta I a seguir, serve para esse propósito:

```
SELECT *
 FROM DM_DOCENTE
  WHERE CO_DOCENTE_IES=850516;
```

Essa consulta demorou **2,5** segundos para ser executada.

Agora, criaremos uma tabela chamada DM_DOCENTE_2, contendo os mesmos registros de DM_DOCENTE, no entanto com os registros ordenados pela coluna CO_DOCENTE_IES, conforme código SQL a seguir:

```
/*
 Tabela DM_DOCENTE_2 com registros ordenados por
CO_DOCENTE_IES;
*/
CREATE TABLE DM_DOCENTE_2 AS
SELECT *
FROM DM_DOCENTE
ORDER BY CO_DOCENTE_IES;
```

Adicionaremos chave primária à tabela DM_DOCENTE_2, escolhendo a coluna CO_DOCENTE_IES, conforme código SQL a seguir:

```
/*
 Ao adicionar chave primária na tabela DM_DOCENTE_2, o SGBD
cria um índice para a coluna CO_DOCENTE_IES.
*/
ALTER TABLE DM_DOCENTE_2 ADD PRIMARY KEY (CO_DOCENTE_IES);
```

Finalmente, executaremos a consulta II:

```
SELECT *
FROM DM_DOCENTE_2
WHERE CO_DOCENTE_IES=850516;
```

Essa consulta demorou **0,06** segundos para ser executada.

O resultado do processamento das consultas é igual, uma vez que temos os mesmos registros em ambas as tabelas. Contudo, a consulta 2 foi processada com mais eficiência.

Após breve contextualização envolvendo consulta, transação e indexação, passaremos a estudar os fatores que influenciam no projeto de bancos de dados relacionais.

Projeto físico em bancos de dados relacionais

Projetar um banco de dados é um processo que envolve as seguintes etapas:

- Levantamento de requisitos
- Projeto conceitual
- Projeto lógico
- Projeto físico

Projetar fisicamente um banco de dados é o mesmo que “colocar a mão na massa”, ou seja, acessar recursos do sistema gerenciador de banco de dados (SGBD) para atividades de criação da estrutura física do banco, que, na maioria das vezes, ocorre com o auxílio de alguma ferramenta CASE capaz de gerar códigos na linguagem SQL para criar as tabelas, relacionamentos e demais componentes do banco de dados.

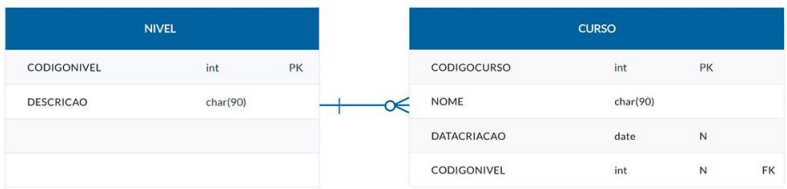
É comum que haja mais de uma alternativa para implementar um banco de dados tomando como base o esquema conceitual. Ainda, o projeto físico de banco de dados é comumente influenciado pelos seguintes fatores:

Consultas e transações de banco de dados

É necessário planejar as consultas e transações que deverão ocorrer no banco de dados. De um modo geral, para cada consulta de recuperação de dados, é necessário conhecer:

- As tabelas acessadas pela consulta.
- As colunas que serão utilizadas em condições de seleção.
- A natureza da condição de seleção: intervalo, igualdade ou desigualdade.
- Colunas utilizadas na composição de operações de junção.
- Colunas cujos valores aparecerão nos resultados da consulta.

Em se tratando de tabelas de um banco de dados, é comum criarmos índices para determinadas colunas. Em especial, colunas relacionadas aos itens 2 e 4 são boas candidatas para serem indexadas.



NIVEL e CURSO

Observe a representação textual a seguir:

```
SELECT *
FROM CURSO
WHERE NOME='Medicina' OR NOME='Nutrição';
```

O SGBD precisa avaliar se há algum registro na tabela CURSO cujo conteúdo da coluna NOME seja “Medicina” ou “Nutrição.” Trata-se de uma consulta enquadrada no item 2: há uma condição de seleção na cláusula WHERE envolvendo a coluna NOME: uma boa candidata para criação de índice.

Veja a consulta II a seguir, que objetiva recuperar o nome do curso e o nível ao qual ele pertence:

```
SELECT NOME, DESCRICAO
FROM CURSO JOIN NIVEL ON (CURSO.CODIGONIVEL=NIVEL.CODIGONIVEL);
```

A consulta usa um comando de junção (JOIN). A condição (CURSO.CODIGONIVEL=NIVEL.CODIGONIVEL) será avaliada diversas vezes ao longo do processamento da consulta. Trata-se de uma consulta enquadrada no item 4: as colunas CODIGONIVEL presentes na tabela são boas candidatas para criação de índices.

No caso de operações de atualização de dados, é necessário conhecer:

- As tabelas-alvo da atualização.
- A categoria da atualização em cada tabela: exclusão, atualização ou inserção.
- Colunas utilizadas em condições de seleção para exclusão ou atualização.
- Colunas-alvo das operações de atualização.

Ainda no contexto de indexação, colunas relacionadas ao item 3 são boas candidatas para serem indexadas. Ao mesmo tempo, o ideal é não criar índices para as colunas relacionadas ao item 4. Vamos estudar um exemplo?

Veja a consulta III a seguir, que objetiva excluir todos os cursos que tenham a **string** “Engenharia”.

```
DELETE FROM CURSO
WHERE NOME LIKE '%ENGENHARIA%';
```

O SGBD precisa localizar os registros para então apagá-los do banco de dados. Para tanto, executará a condição de seleção presente no WHERE. Trata-se de uma consulta enquadrada no item 3: a coluna NOME presente na tabela é boa candidata para criação de índice.

Frequência de chamada de consultas e transações esperada

Vimos a importância de identificar detalhes sobre as consultas de recuperação e transações de atualização esperadas. No entanto, saber a respeito da frequência de uso esperada para operações de consulta e transações também é uma boa estratégia para obter desempenho.

Atenção

Aplicando-se a “regra do 80/20”, conhecida como Princípio de Pareto, em um sistema de banco de dados, estima-se que 80% do processamento é originado de somente 20% das consultas e transações. Por isso, é rara a necessidade de coletar informações estatísticas completas e taxas de chamada para todas as consultas e transações, bastando priorizar 20% das mais relevantes.

Restrições de tempo de consulta e transações

Dependendo da natureza da aplicação, podem existir consultas e transações com restrições de desempenho bastante rigorosas. Para exemplificar, poderia existir a restrição de que uma transação de compras tenha que terminar o seu processamento de pagamento dentro de sete segundos em 90% das vezes em que é chamada, e que ela nunca deve ultrapassar quinze segundos.

Essas restrições referentes ao tempo têm forte impacto nas colunas candidatas a serem indexadas. Em especial, tais colunas devem ser priorizadas quando da decisão da criação de índices para as tabelas.

Frequências esperadas de operações de atualização

Se a tabela é atualizada com frequência, deve-se evitar a criação de índices nas colunas, pois a atualização de colunas indexadas, frequentemente, requer atualização na estrutura de índice.

Exemplo

Se uma tabela possui cinco colunas indexadas, a inserção de um novo registro requer a atualização dos índices, o que pode causar lentidão nesse tipo de operação.

Restrições de exclusividade em colunas da tabela

É útil criar índice para cada coluna com restrição de unicidade na tabela. Em uma operação típica de inserção, o SGBD pode validar essa restrição de exclusividade fazendo consulta à estrutura de índice, rejeitando a inserção caso o valor da coluna seja encontrado no índice.

Atividade 2

Considerando um banco de dados relacional, avalie as alternativas a seguir que tratam de vantagens da indexação.

I. Acelerar a recuperação de dados em consultas.

II. Reduzir o espaço em disco ocupado pelo banco de dados.

III. Melhorar a performance de consultas com filtros e ordenações.

IV. Permitir a criação de consultas complexas com várias condições.

Está correto o que se afirma em:

A

I, II, III e IV.

B

II, III e IV.

C

I, III e IV.

D

I, II e IV.

E

I, II e III.

A alternativa C está correta.

As vantagens da indexação em banco de dados são: aceleração da recuperação de dados, melhoria da performance de consultas com filtros e ordenações, facilitação da criação de consultas complexas.

A indexação não reduz o espaço em disco ocupado pelo banco de dados. Na verdade, ela pode aumentar o espaço ocupado, pois cria estruturas adicionais para armazenar os índices.

Prática de instalação de SGBD e criação de banco de dados

Vamos criar um banco de dados de documentos!

Neste vídeo, você vai praticar a instalação do SGBD PostgreSQL e a criar um banco de dados.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Roteiro de prática

Gere o script do banco dos documentos.

1. Acesse o brmodelo3 e abra o arquivo DOCUMENTOLOGICO.brM3.
2. Gere o modelo físico.
3. Salve o script.

O arquivo de script está disponível para download: criadoc.sql.

Para instalar o SGBD PostgreSql, acompanhe os seguintes passos.

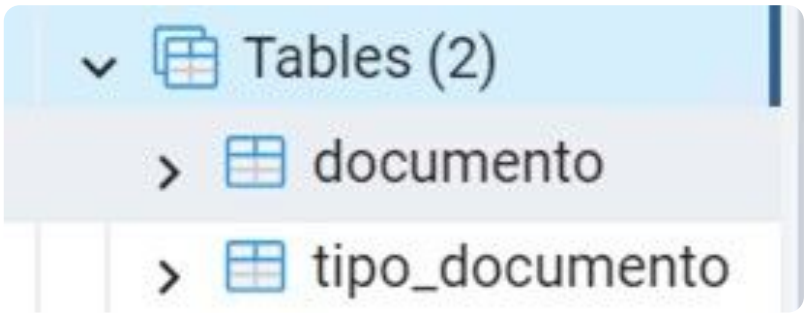
1. Vá para o site oficial do PostgreSQL em https://www.postgresql.org/.
2. Selecione a versão mais recente compatível com o seu sistema operacional (Windows) e faça o download do instalador.
3. Execute o arquivo baixado para iniciar o instalador. Siga as instruções na tela para configurar o PostgreSQL.
4. Durante a instalação, será solicitada a configuração de um cluster. Defina uma senha para o superusuário (postgres) e configure as opções conforme necessário.
5. Escolha a porta padrão (geralmente 5432) e os diretórios para armazenamento de dados e arquivos de configuração.
6. Complete o processo de instalação e, se necessário, reinicie o sistema.
7. Instale a ferramenta de administração PgAdmin4.

Para criar o banco de documentos:

1. Abra o PGAdmin4 e faça conexão no servidor.
2. Crie um banco de dados chamado documentos.

3. Abra uma janela de consulta.
4. Carregue o script na janela de consulta.
5. Execute o script.
6. Valide a criação das tabelas na aba Tabelas do banco de documentos.

Observe na imagem a seguir.



Visualização das tabelas usando o pgAdmin.

Para inserir os dados nas tabelas, rode o script populadoc.sql.

Atividade 3

Você foi designado para configurar um ambiente de desenvolvimento em um sistema Windows e precisa instalar e configurar o PostgreSQL como parte desse processo. Além da instalação do PostgreSQL, a equipe de desenvolvimento está explorando a utilização do Application Stack Builder (ASB) no PostgreSQL. Explique o propósito e a funcionalidade ASB nesse contexto.

Chave de resposta

O Application Stack Builder (ASB) é uma ferramenta gratuita e de código aberto que auxilia na instalação e no gerenciamento de extensões para o PostgreSQL. Ele oferece uma interface gráfica amigável e simplifica os processos seguintes.

1. Instalar extensões: Fornece uma lista completa de extensões disponíveis para o PostgreSQL, incluindo suas funcionalidades e requisitos, permitindo a instalação com um único clique.
2. Gerenciar extensões: Permite visualizar informações detalhadas sobre as extensões instaladas.
3. Configurar extensões: Fornece uma interface para configurar as opções de configuração das extensões instaladas.

Prática de consultas envolvendo mais de uma tabela e indexação

Vamos completar nosso banco de documentos com a criação de mais duas tabelas. Além disso, vamos analisar o uso de índices e como eles podem melhorar as consultas SQL.

Neste vídeo, você vai praticar consultas envolvendo mais de uma tabela e indexação.

Conteúdo interativo

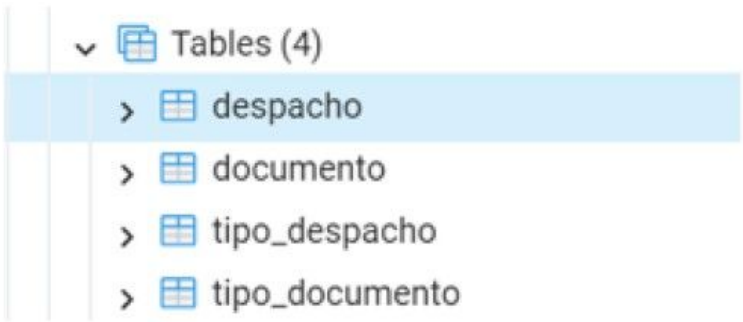
Acesse a versão digital para assistir ao vídeo.

Roteiro de prática

Para complementar o banco de documentos:

1. Abra o PGAdmin4 e faça conexão no servidor.
2. Selecione o banco de dados de documentos.
3. Abra uma janela de consulta.
4. Carregue o script na janela de consulta.
5. Execute o script.
6. Valide a criação das tabelas na aba Tabelas do banco de documentos.

Observe na imagem a seguir.



Visualização das tabelas usando o pgAdmin

Para complementar o banco de documentos, rode o script criapopdespacho.sql.

Para ver o uso de índices:

1. Realize consultas na tabela despacho, pesquisando por determinado documento.
2. Faça junções entre documento de despacho.
3. Crie um índice na coluna documento da tabela despacho.
4. Realize novamente as consultas e analise se o tempo de resposta diminuiu.

Atividade 4

Considere um banco de dados de uma loja on-line que armazena informações sobre produtos, clientes e pedidos. O banco de dados contém milhões de registros e é fundamental para a eficiência operacional da empresa. Os desenvolvedores da empresa estão otimizando as consultas SQL para garantir um desempenho ideal do sistema. Durante a revisão do código, surge a discussão acerca da importância dos índices em consultas SQL e como eles podem afetar o desempenho das consultas. Os desenvolvedores estão debatendo sobre as melhores práticas para melhorar a velocidade de busca e a recuperação de dados sem comprometer a integridade dos resultados. Desse modo, surge a seguinte questão: Qual é a principal vantagem de utilizar índices em consultas SQL?

A

Redução do número de linhas retornadas pela consulta.

B

Aumento da complexidade da consulta.

C

Melhoria no desempenho da consulta.

D

Aumento do tempo de execução da consulta.

E

Nenhuma diferença perceptível.

A alternativa C está correta.

Os índices são estruturas de dados criadas em colunas de tabelas SQL para acelerar a recuperação de registros durante consultas. Ao criar um índice em uma coluna específica, o banco de dados cria uma estrutura de dados adicional que permite localizar registros rapidamente com base nos valores nessa coluna. Isso resulta em uma melhoria significativa no desempenho das consultas, pois o banco de dados pode acessar os registros desejados de forma mais eficiente. Portanto, ao utilizar índices em consultas SQL, é possível reduzir o tempo de execução e melhorar a eficiência geral das operações de busca.

Desnormalizar para ganhar desempenho

Este núcleo explora a técnica de desnormalização em bancos de dados relacionais, revelando como ela pode ser utilizada para otimizar o desempenho de consultas e aumentar a eficiência do sistema. Vamos abordar os princípios e as diferentes técnicas de desnormalização e seus impactos no desempenho, na escalabilidade e na manutenção do banco de dados.

Com este vídeo, você verá como a desnormalização em bancos de dados relacionais pode otimizar o desempenho de consultas e aumentar a eficiência do sistema.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Quando um esquema de banco de dados está normalizado até a 3FN, os problemas com redundância de dados são minimizados, pois, em geral, existe uma tabela para cada objeto modelado.

Ao mesmo tempo, vimos que processar a consulta anterior requer acesso às duas tabelas para recuperar as informações – o que gera um custo adicional de processamento.

> Se quisermos priorizar desempenho, teremos que sacrificar as vantagens de um modelo normalizado. Esse processo é conhecido por desnormalização.

Nossa intenção a partir de agora é gerar uma estrutura que permita obter os mesmos resultados da consulta anterior, no entanto, usando somente uma tabela. Esse tipo de situação é comum quando temos a necessidade de produzir relatórios em um sistema.

Ao desnormalizar o modelo, ficamos com a seguinte tabela: DM_DOCENTE_2 (CO_DOCENTE_IES, CO_IES, NO_IES, CO_MUNICIPIO_NASCIMENTO, NOME). Note que a coluna NOME é dependente da coluna CO_MUNICIPIO_NASCIMENTO, ou seja, precisamos ter em mente que estamos diante de uma dependência funcional parcial, violando a 2FN.

Diante da nova estrutura, o código a seguir recupera as informações, agora envolvendo somente uma tabela:

```
SELECT CODIGO_DOCENTE_IES, NOME
 FROM DM_DOCENTE_2;
```

O processo de desnormalização deve ser planejado com critério, visto que, ao mesmo tempo em que há potencial de ganho em relação ao desempenho de determinadas consultas, a desnormalização introduz redundância nos dados e, ao mesmo tempo, é necessária atualização adicional visando manter a consistência das colunas redundantes.

Ao longo deste módulo, percebemos que a criação do modelo físico em um SGBD está atrelada ao objetivo de criar um banco de dados de maneira que problemas de baixo desempenho sejam evitados. Para isso, é necessário mapear as principais consultas e transações a serem processadas ao longo do ciclo de vida do banco de dados.

Atividade 5

Em um banco de dados relacional, qual das alternativas a seguir não é uma vantagem da desnormalização?

A

Acelerar o tempo de resposta de consultas frequentes.

B

Reduzir a redundância de dados, diminuindo o espaço em disco ocupado.

C

Melhorar a escalabilidade do banco de dados em grandes conjuntos de dados.

D

Simplificar as consultas complexas, tornando-as mais fáceis de escrever e de entender.

E

Eliminar a necessidade de joins em consultas que envolvem várias tabelas.

A alternativa B está correta.

As vantagens da desnormalização são: acelerar o tempo de resposta de consultas frequentes; melhorar a escalabilidade do banco de dados em grandes conjuntos de dados; e simplificar as consultas complexas, tornando-as mais fáceis de escrever e de entender.

A desnormalização não reduz a redundância de dados; pelo contrário, ela aumenta a redundância para otimizar o desempenho. Isso pode levar a um aumento do espaço em disco ocupado pelo banco de dados.

Considerações finais

O que você aprendeu neste conteúdo?

- O modelo relacional.
- O processo de normalização.
- As formas normais: 1FN, 2FN e 3FN.
- O mapeamento conceitual-lógico.
- Consultas em uma ou mais tabelas.
- Indexação e transação em um banco de dados.
- O processo de desnormalização.

Explore +

- Para aprofundar seu conhecimento sobre a ferramenta BrModelo, acesse o portal GitHub e obtenha informações sobre a correção de bugs e outras funcionalidades.

- Caso tenha interesse em continuar estudando as ferramentas comerciais, acesse o site Vertabelo. Observe como funcionam as ferramentas de modelagem, as quais permitem o uso de alguma notação gráfica para representar um banco de dados relacional, conforme mencionado na seção **Esquema diagramático de banco de dados relacional**.

- Para complementar seu estudo, leia o texto *Edgar Frank Codd e o Banco de Dados Relacional: uma contribuição para a História da Computação*, de Odécio Souza, publicado pela Pontifícia Universidade Católica de São Paulo, São Paulo, 2015. O material fala sobre o modelo relacional que foi introduzido por Ted Codd, pesquisador visionário que apresentou em 1970 as bases científicas sobre as quais a maior parte dos SGBDs relacionais fazem uso. Trata-se de um trabalho completo sobe as contribuições de Codd para a Ciência da Computação.

- Recomendamos ainda que você leia na referência (Elmasri; Navathe, 2019) o capítulo que trata sobre indexação. O desempenho de um banco de dados relacional em termos de recuperação de informações a partir de determinada consulta tem relação direta com o projeto dos mecanismos de indexação.

Referências

ELMASRI, R.; NAVATHE, S. **Sistemas de Banco de Dados**. 7. ed. São Paulo: Pearson, 2019.

HEUSER, Carlos A. **Projeto de Banco de Dados**. 6. ed. Porto Alegre: Bookman, 2009.

SOUZA, Odécio. **Edgar Frank Codd and the Relational Database: a contribution to the History of Computing**. 2015. 155 f. Dissertação (Mestrado em História da Ciência) - Pontifícia Universidade Católica de São Paulo, São Paulo, 2015.