Banco de Dados - Ciclo 2 - Webaula 16

FURB

WEBAULA 16

- Projeto Físico de Dados
- Referências

QUESTION BANKS

Projeto Físico de Dados

Nas seções anteriores tivemos a oportunidade de conhecer a modelagem “entidade-relacionamento” (MER) e a normalização de dados, ambas técnicas da análise de dados que compõem o modelo conceitual de um banco de dados. Agora vamos abordar o conceito e as características do projeto físico de banco de dados e conhecer as técnicas para transformar a MER, resultado do modelo conceitual, em um projeto físico de banco de dados. Também, abordaremos aspectos relevantes sobre a derivação de um modelo orientado a objetos (diagrama de classes) para o projeto físico do banco de dados. O projeto físico é a última etapa antes da consolidação das estruturas de dados projetadas em estruturas internas (físicas) do banco de dados (tabelas e restrições de integridade).

Antes, convém abordarmos brevemente o conceito de projeto lógico. O projeto lógico ou também conhecido como modelo lógico, é a etapa intermediária abstrata na qual as definições conceituais representadas graficamente no MER são transformadas em estruturas físicas através do projeto físico de dados. Neste contexto podemos afirmar que o projeto lógico amplia a atenção do analista/projetista para os atributos e seus tipos de dados assim como as regras de negócio do sistema, sobre os quais os dados serão submetidos. O resultado desta etapa é o esquema lógico de dados. Tomemos como exemplo projetos de banco de dados produzidos por uma ferramenta CASE, como exemplo a DB Desinger Fork. Nessa ferramenta os atributos já estão vinculados aos tipos de dados, índices, restrições entre outras propriedades do MYSQL, Pois bem, podemos afirmar então que a ferramenta CASE DB Designer Fork trabalha com o projeto lógico e seu resultado é um esquema ou modelo lógico de dados para o MySQL.

Portanto, o modelo lógico é abstrato, ou seja, ocorre implicitamente no momento que definimos qual SGBD irá comportar as estruturas definidas no MER. A seguir abordamos a etapa do projeto físico do banco de dados, começando pela derivação a partir de um MER.

Derivação a partir do MER

O projeto físico dos dados é a etapa na qual as estruturas conceituais apuradas e representadas através do MER são criadas através dos princípios do modelo relacional e das características do SGBD escolhido. Portanto, é fundamental que você saiba que o projeto físico resultante de um MER pode apresentar características distintas em relação ao resultado da transformação do mesmo MER para outro SGBD. Isso ocorre, por exemplo, pelas diferenças entre os tipos de dados e índices utilizados. Atualmente boa parte das ferramentas CASE, quando necessário, realizam a transformação do MER para o projeto físico do banco de dados definido. Independentemente disso, você terá a oportunidade conhecer as principais regras de derivação do MER para o projeto físico.

A 1ª regra indica que cada entidade do MER gerará um tabela no projeto físico. Esta é a regra mais simples e básica do processo de derivação. A entidade é a unidade básica de armazenamento no projeto físico, que passa a se chamar de tabela. Observe a seguir a aplicação desta regra na figura 27.

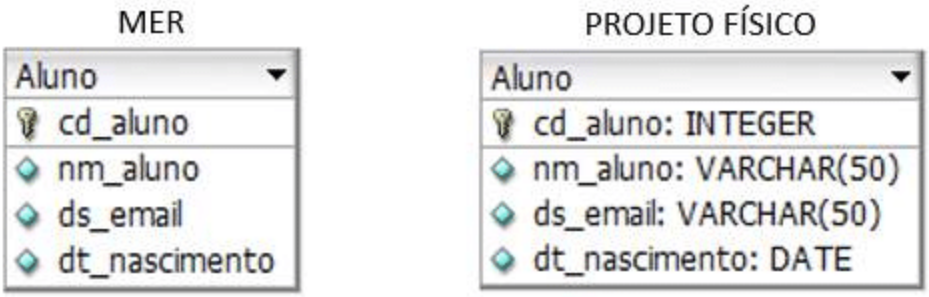


Figura 27

Dica

As ilustrações acima, bem como as demais que você verá nos próximos exemplos, foram geradas na ferramenta CASE DBDesigner Fork.

Cada subentidade também gerará uma entidade, cuja chave primária é a chave da superentidade correspondente. Neste caso o dado que identifica o tipo de subentidade deve ser incluído na tabela da superentidade como se fosse uma chave estrangeira. Observe a aplicação desta 2ª regra através da ilustração da figura 28.

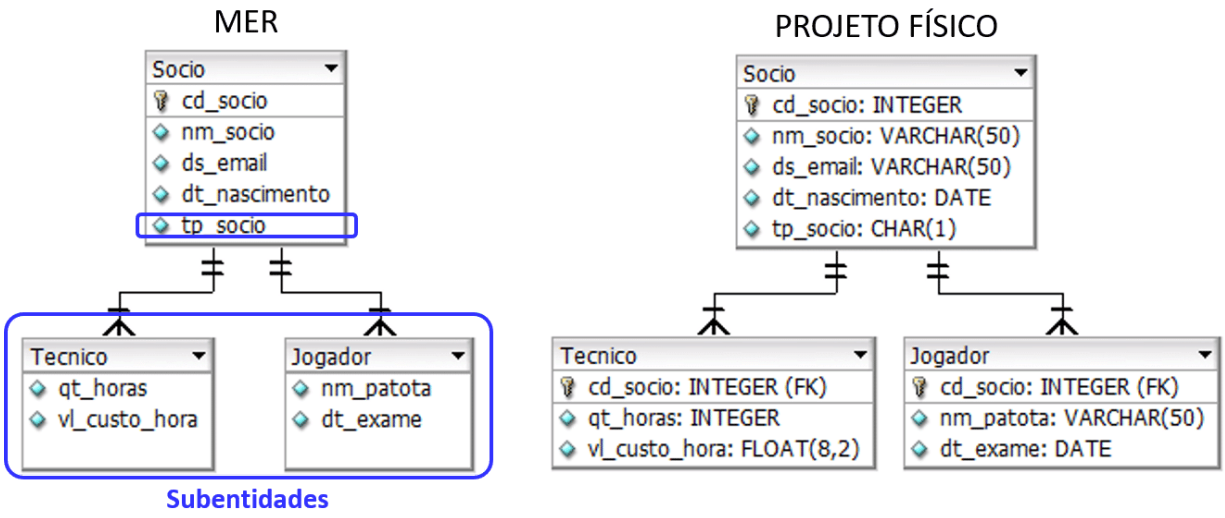


Figura 28

A entidade fraca, com dependência da existência, também gera uma tabela, cuja chave primária é a concatenação da chave primária da entidade forte com a chave primária da entidade fraca. Observe a aplicação desta 3ª regra através da ilustração da figura 29.

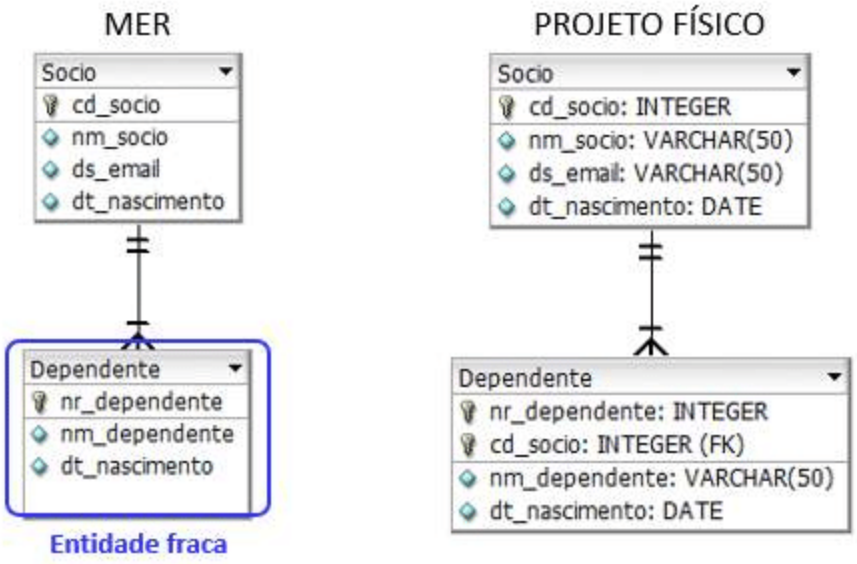


Figura 29

Outra regra (4ª) que de alguma forma você já deve a oportunidade de conhecer está associada ao relacionamento com conectividade muitos-para-muitos. Nesta, a estrutura resultante do relacionamento, necessariamente, deve gerar uma nova tabela, cuja chave primária será a concatenação das chaves primárias das entidades relacionadas. Confira a aplicação desta regra (figura 30).

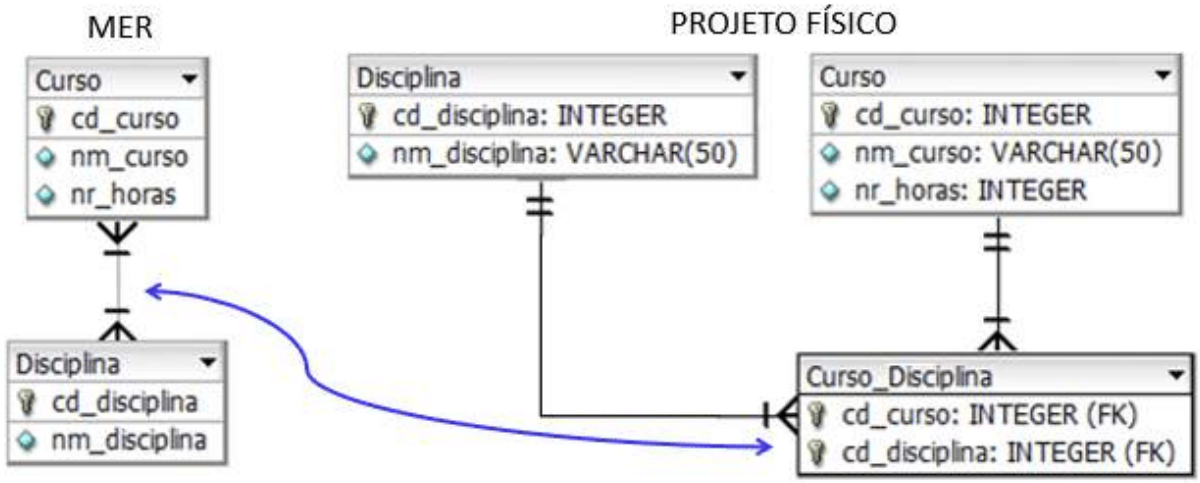


Figura 30

Como estamos tratando da derivação envolvendo relacionamentos, passamos agora para a regra (5ª) envolvida no relacionamento com conectividade um-para-muitos. Nesta devemos incorporar junto à tabela resultante da entidade que ocupa o lado “n” (muitos) do relacionamento uma chave estrangeira equivalente à chave primária da tabela do lado “1” (um). Confira a aplicação desta regra ilustrada na figura 31.

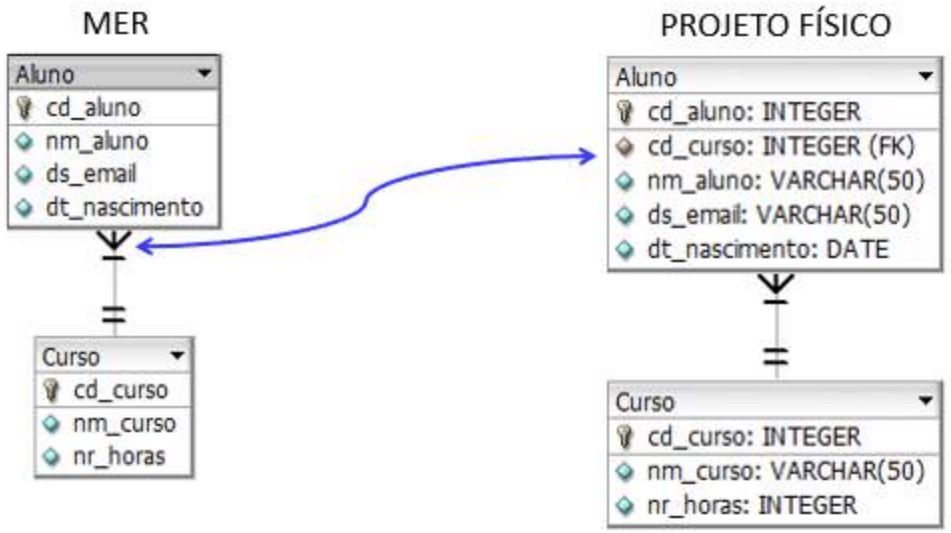


Figura 31

Temos, ainda, a derivação envolvendo o relacionamento de conectividade um-para-um. Neste é necessária a inclusão de uma chave estrangeira equivalente à chave primária da outra entidade que resultará em tabela. Neste caso fica a critério do analista/projetista a escolha do lado em que receberá esta chave. Uma dica é observar a estrutura que demandará menor volume de dados. Ela será uma séria candidata a receber o elemento de ligação (chave). Confira a aplicação desta regra (6ª) ilustrada na figura 32.

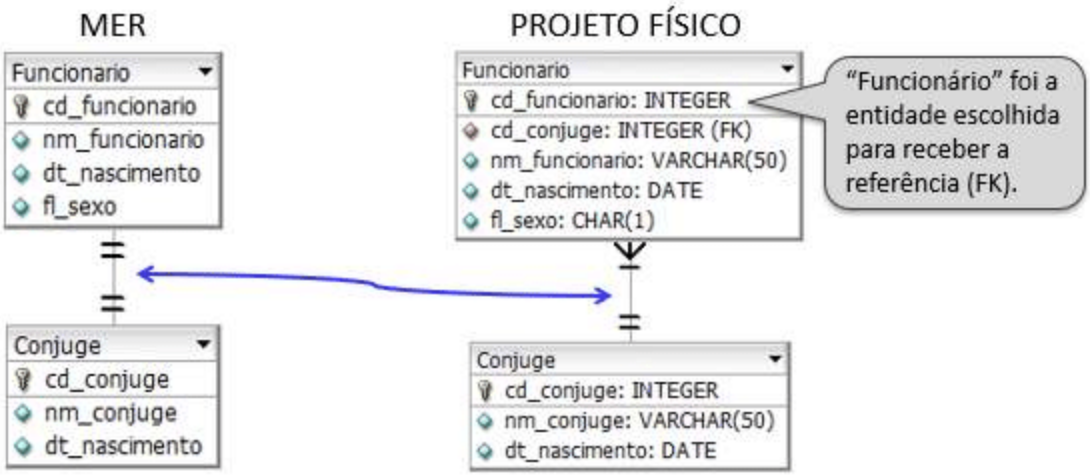


Figura 32

Dica

Observe que a ilustração gerada a partir da ferramenta CASE DBDesigner em que temos um relacionamento de conectividade um-para-um é transformado, automaticamente, em relacionamento de conectividade um para muitos. Na prática isso ocorre pela necessidade da indicação de qual tabela irá receber o atributo de ligação. Neste momento, podemos concluir que, implicitamente, o ocorreu uma mudança conceitual da conectividade do relacionamento.

Por fim destacamos ainda que ao utilizar a ferramenta CASE DBDesigner Fork você poderá alternar a visualização do modelo de dados entre conceitual e físico com facilidade, visto que ela faz essa transformação de forma automática para você. Lembre-se, na ferramenta CASE DBDesigner, por padrão, todo projeto é construído tendo como SGBD o MySQL.

Ao longo da sua trajetória no curso você pôde perceber que é cada vez maior a especificação de sistemas computacionais utilizando os conceitos das orientações a objetos. É possível então que você se pergunte: por que não utilizamos a Linguagem de Modelagem Unificada (UML) para a definição da estrutura de dados que será persistida? Existem literaturas que apontam para esse caminho, mas ele

ainda é pouco explorado. E por que isso ocorre? A explicação é simples. A representação conceitual de um modelo de dados, através de um diagrama de classes, por exemplo, é insuficiente quanto às reais estruturas a serem derivadas para o modelo de banco de dados relacional. É justamente isso que vamos explorar a partir de agora. A seguir vamos indicar, por meio de exemplos, algumas diretrizes que poderão ajudá-lo no momento da transformação do diagrama de classes em um modelo "entidade-relacionamento" (MER).

É importante ressaltar que nosso foco agora não é discutir a modelagem do diagrama de classe, nem tão pouco da especificação através da UML. Vamos utilizar o diagrama de classes como modelo orientado a objetos que desejamos transformar em modelo para banco de dados relacional.

Vamos conhecer agora algumas diretrizes básica para a possamos derivar um diagrama de classe para um projeto de banco de dados relacional. A primeira indica que uma classe se torna uma tabela. A partir desta definição temos que:

- o atributo de uma classe torna-se uma coluna em uma tabela;
- o tipo de atributo de uma classe torna-se um tipo de coluna em uma tabela;
- se o atributo de uma classe indicar marcação {nullable}, a coluna da tabela poderá ser nula, do contrário será de preenchimento obrigatório;
- se o atributo de uma classe tiver valor inicial, a coluna na tabela deverá apresentar a propriedade default;
- caso não haja definição de atributos identificadores (únicos) em classes sem generalização, deve-se criar uma coluna que servirá de chave primária da tabela;
- se o atributo possuir marcador {alternate}, deve-se usar a restrição de UNIQUE KEY na tabela;

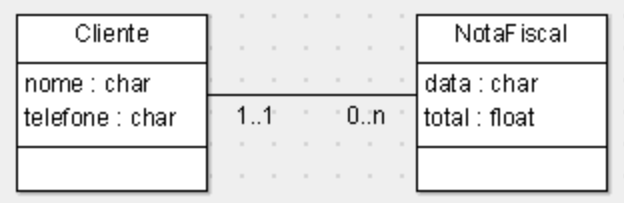
Dica

Alguns conceitos apresentados aqui você verá na prática quando conhecer em detalhes a linguagem SQL.

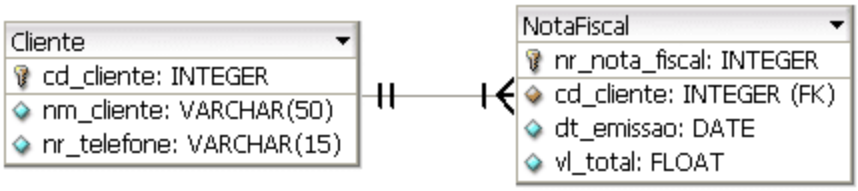
Dando sequência às diretrizes, temos os cenários que envolvem classes e subclasses, ligações através de associações por sinal, a mais utilizada representação de relações entre classes de dados e que você irá encontrar com maior frequência:

- nos casos envolvendo classes e subclasses, deve-se acrescentar uma coluna que servirá como elo entre as tabelas;
- acrescentar a cláusula check para restrições explícitas;
- criar chaves estrangeiras em classes com cardinalidade 0..1 e 1..1 na associação;
- criar chave primária para agregação composta com chave estrangeira para a tabela agregada;
- otimizar classes de associação binária;
- criar tabelas para associação n..n e ternárias;
- criar as restrições de chaves primárias e estrangeiras nas associações n..n e ternárias.

Considere parte de um cenário em que temos uma classe “nota fiscal” e uma classe “cliente”. Entres elas há uma relação de associação. A representação do diagrama de classes pode ser vista na figura 33.



Ao aplicarmos as diretrizes para transformação para um projeto físico de dados temos a representação gráfica da figura 34.



Dica

A associação no diagrama de classes (UML) é equivalente ao relacionamento no modelo “entidade-relacionamento”.

Seguimos agora com um exemplo envolvendo composição. Considere o exemplo envolvendo “nota fiscal” e “item de nota fiscal”. A representação do diagrama de classes correspondente é apresentada na figura 35.

A aplicação das diretrizes para transformação de uma composição para um projeto físico de dados é representada da figura 36.

No exemplo acima observamos que foi criada uma chave primária na tabela “nota fiscal”. Esta, por sua vez, foi utilizada como chave estrangeira para servir de ligação com a tabela “item nota fiscal”, assim como parte da chave primária desta última tabela. É fundamental ressaltar que, possivelmente, outros atributos possam ser adicionados a tabela “item nota fiscal”, inclusive, como parte da chave primária.

Encerramento

Ao concluir este ciclo você teve a oportunidade de conhecer conceitos, princípios básicos e técnicas para construção de modelos entidade-relacionamento (MER). A partir do MER você foi apresentado as diretrizes para derivar um modelo para um projeto físico de banco de dados relacional.

Em dado momento, brevemente, falamos no uso de ferramenta CASE. As ferramentas CASE são parte do processo de construção de modelos de dados. Elas estarão inseridas nas atividades práticas da disciplina. Aliás, serão muitas as atividades práticas.

Assim como não se aprende programação lendo, exclusivamente, livros ou mesmo assistindo vídeos nas redes sociais, modelar projetos de bancos de dados também requer experimentação. Por meio do exercício constante e intensivo você irá potencializar o desenvolvimento de aptidões para o sucesso na atividade.

CONTINUE

Referências

Referências

ALVES, William Pereira. Banco de dados. São Paulo: Erica, 2014

CARDOSO, Vírginia M. Sistemas de banco de dados. São Paulo : Saraiva, 2008.

CHEN, Peter. Gerenciando banco de dados: a abordagem entidade-relacionamento para projeto lógico. São Paulo, McGraw-Hill, 1990.

HEUSER, Carlos Alberto. Projeto de banco de dados. Porto Alegre: Bookman, 2011.

MACHADO, Felipe Nery Rodrigues. Banco de dados: projeto e implementação.3. São Paulo : Erica, 2014.

Referências de Imagens

Divisão de Modalidades de ensino (DME), Fundação Universidade Regional de Blumenau (FURB), 2019.