Pedro F. Carvalho
Analista de Sistemas
contato@pedrofcarvalho.com.br
www.pedrofcarvalho.com.br

ORACLE Certified Professional

Microsoft CERTIFIED Technology Specialist

ITIL V3 ITIL V3 Foundation

Dicas de modelagem de dados

Modelagem de dados: projeto conceitual

1. Sempre faça modelagem de dados, isso ajuda no entendimento do problema e no planejamento de uma solução mais aderente aos seus objetivos. O objetivo do modelo conceitual é a definição do problema, não da solução.

2. Existem diferentes terminologias para modelagem conceitual. Os exemplos da Figura 1 representam diferentes notações para um relacionamento 1:N, sendo as mais comuns de serem implementadas pelas atuais ferramentas CASE. A primeira notação, de Peter Chen, é a mais utilizada para a construção do modelo conceitual enquanto a segunda, do Pé de Galinha, é a mais utilizada para a construção do modelo lógico.

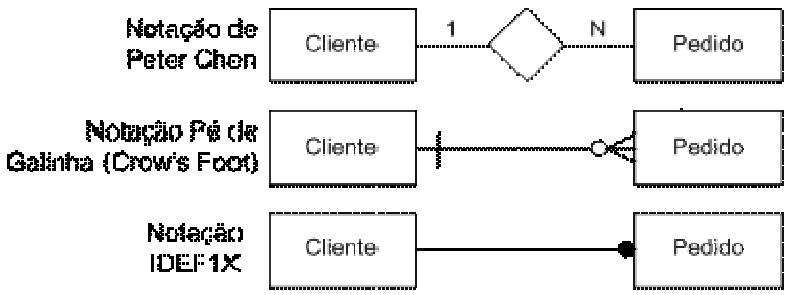

Figura 1. Notações para modelagem de dados.

3. Elimine qualquer redundância de dados. Redundâncias permitidas são aquelas relativas a chaves estrangeiras, que fazem referência à chave primária de outra tabela. Por exemplo, não se deve repetir o nome do cliente em seus pedidos, pois ele pode ser facilmente obtido através do relacionamento.

4. Utilize um padrão para dar nomes a entidades. Normalmente, nomes de entidades são no singular.

5. Dê nomes significativos a entidades, atributos e relacionamentos. Nomes que não representam seu real objetivo dificultam a compreensão do modelo.

6. Deve-se determinar os relacionamentos e, decidir como é a relação de dependência entre cada entidade é sempre importante. Os tipos de relacionamento podem ser:

· 0:1 (mínimo: nenhum – máximo: um);

· 0:N (mínimo: nenhum – máximo: muitos);

· 1:1 (mínimo: um – máximo: um);

· 1:N (mínimo: um – máximo; muitos);

· N:N (mínimo: muitos – máximo: muitos).

7. Relacionamentos não obrigatoriamente precisam ter nomes, mas é uma boa prática de modelagem nomeá-los, pois facilita o entendimento. Assim, utilize nomes significativos para os relacionamentos, como apresentado na Figura 2 para o relacionamento Lotação.

Figura 2. Nomeando relacionamentos.

8. Relacionamentos N:N ou que possuem atributos normalmente geram novas tabelas no modelo lógico. O nome do relacionamento pode ser utilizado como nome da tabela e deve ser cuidadosamente escolhido, como exemplificado na Figura 3.

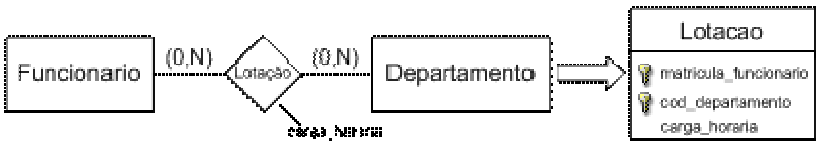

Figura 3. Gerando tabelas a partir de relacionamentos.

9. Ao dar nomes a atributos determinantes, coloque sempre algo que identifique a entidade que está sendo relacionada. Por exemplo, um atributo chave cod_cliente é melhor do que simplesmente código. Lembre-se que, no modelo lógico, este atributo será repetido como chave estrangeira nas entidades relacionadas. Percebe-se na Figura 4 que, se o nome da chave primária da entidade Cliente for apenas codigo, quando repetida na entidade Pedido para representar o relacionamento, seu nome não será representativo, ou seja, não se pode observar facilmente a qual entidade está associado. Se o atributo tivesse sido chamado de cod_cliente, esta associação seria muito mais fácil. Este problema seria mais grave se a entidade Pedido estivesse associada à outra cujo nome de sua chave primária também fosse codigo. Teríamos mais de um atributo com o mesmo nome, que as ferramentas CASE costumam modificar atribuindo uma seqüência numérica, o que dificulta enormemente a compreensão do modelo.

Figura 4. Nomeando atributos chave.

10. Procure padronizar os nomes dos atributos de seu diagrama entidade relacionamento. Por exemplo, ao ver o atributo num_pedido percebe-se facilmente que seu tipo é numérico. A padronização garante um bom entendimento dos atributos perante a equipe de desenvolvimento, garantindo eficiência na fase de desenvolvimento da aplicação.

11. Atenção para entidades desconectadas no diagrama. Podem não ser um problema, mas precisam ser verificadas. Por exemplo, é comum que entidades que representem parâmetros estejam soltas no diagrama.

12. Cuidado com relacionamentos com participação total em ambos os lados de um relacionamento (obrigatoriedade dos dois lados). Isso pode provocar uma situação onde uma entidade dependa da outra recursivamente. Além disso, deve-se estar atentos a situações onde o banco de dados ainda está vazio e precisa-se cadastrar registros em apenas uma das entidades. Observe o exemplo abaixo da Figura 5. Um Cliente possui um ou mais Pedidos e um Pedido é obrigatoriamente de um Cliente. Devido à obrigatoriedade em ambos os lados do relacionamento, esta situação impede que Clientes sejam cadastrados sem Pedidos e que Pedidos sejam feitos sem Clientes.

Figura 5. Relacionamento com participação total em ambos os lados.

Modelagem de dados: projeto lógico

13. Muitas vezes costuma-se “pular” a etapa de modelo conceitual de dados e passa-se diretamente para o modelo lógico. Em grande parte pode-se creditar isso a muitas ferramentas CASE, que partem diretamente do modelo lógico. Considera-se isso perigoso. A modelagem conceitual foi criada para atender às necessidades do usuário, sem limitações tecnológicas. Desta forma, no primeiro modelo, não se deve preocupar com chaves primárias e estrangeiras. As chaves são necessárias uma vez que é assim que os SGBDs trabalham, mas isso não faz parte das necessidades do usuário. O mesmo pode ser dito com relação ao tipo e tamanho de dados, e ainda mais do mapeamento de relacionamentos 1:1 e n:n. O objetivo do projeto lógico é a definição da solução.

14. Vale lembrar que nem todo modelo lógico de dados é a cópia fiel do modelo conceitual de dados.

15. Toda entidade do modelo conceitual vira uma tabela no modelo lógico.

16. Deve-se relacionar diretamente cada coluna ao escopo da tabela: se um campo descreve o assunto de uma tabela diferente, este campo deve pertencer à outra tabela.

17. Elimine colunas repetidas no modelo: se uma informação se repete em diversas tabelas, é um indício de que existem colunas desnecessárias. Deve-se buscar a tabela que faça mais sentido para conter a coluna em questão.

18. Dimensione corretamente os tipos de dados em função de seus conteúdos para diminuir o espaço de armazenamento.

19. Campos de texto com tamanho variável tendem a consumir menos espaço de armazenamento.

20. Defina corretamente a obrigatoriedade para atributos das entidades de forma a retratar o objetivo da entidade. Por exemplo, o nome do cliente deve ser obrigatório, pois não faz sentido cadastrar um cliente sem seu nome. Muitas vezes, preocupa-se apenas com a obrigatoriedade para os atributos chave, mas esta questão é importante para todos os atributos, tomando-se o devido cuidado de não impor restrições demais que impeçam que novos registros possam ser inseridos.

21. Elimine atributos derivados ou calculados: não é recomendado armazenar o resultado de cálculos nas tabelas. O correto é que o cálculo seja gerado sob demanda, normalmente em uma consulta.

22. Toda tabela deve ter uma chave primária, que pode ser simples ou composta. A chave primária é o identificador do registro e deve ser única dentro de uma tabela.

23. Ao determinar a chave primária, deve-se escolher, em cada tabela, quais colunas formarão a chave primária. Para uma coluna ser candidata à chave primária, deverá atender aos principais requisitos:

· Deverá ser a menor possível;

· O seu valor deverá ser diferente de vazio ou zero (not null);

· Deverá ser de preferência numérica;

· O seu valor deverá ser único para toda a tabela.

24. Chaves estrangeiras devem corresponder a chaves primárias da relação associada ou ser nulas, quando não for um campo obrigatório.

25. Crie chaves cegas (Blind Key) toda vez que não for possível identificar a chave primária, ou quando esta for muito complexa, composta por muitos atributos. Esses tipos de atributos geralmente são conhecidos por atributos falsos, ou seja, não fazem parte de forma explícita da regra de negócio, porém foram criados para garantir flexibilidade ou integridade ao modelo de dados desenvolvido. Por exemplo, um funcionário pode ter diversos dependentes, que possuem nome e data de nascimento. Nenhum destes atributos, nem mesmo a concatenação deles, garante uma chave primária única. Uma solução é a criação de um novo atributo determinante, como cod_dependente.

26. Refine os relacionamentos uma vez que alguns destes poderão gerar novos elementos que complementarão o modelo, sendo estes:

· Relação de Herança (mutuamente exclusivo ou não): por exemplo, numa hierarquia de Funcionario contendo especializações Gerente e Engenheiro, um determinado funcionário pode ser Gerente e Engenheiro ao mesmo tempo?;

· Entidade Associativa: gerada quando informações que necessitam ser armazenadas não podem ser colocadas numa das entidades do relacionamento. Acontece em situações de relacionamentos com atributos ou relacionamentos N:N.

27. Relacionamentos são representados por chaves estrangeiras, ou Foreign Key (FK), atributos correspondentes à chave primária de outra relação, estabelecendo a base para a integridade referencial. No exemplo da Figura 6, o atributo cod_cliente, chave primária da tabela Cliente, foi repetido na tabela Pedido para representar o relacionamento. Desta forma, um Pedido possui um cliente que necessariamente deve estar presente na tabela Cliente (através de seu código).

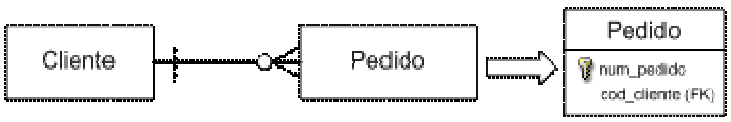

Figura 6. Chaves estrangeiras representando relacionamentos.

28. Relacionamentos 1:1 podem ser mapeados numa única tabela (quando possuem a mesma chave primária), em duas tabelas (quando as chaves primárias são diferentes e um dos lados do relacionamento é obrigatório) ou em três tabelas (quando o relacionamento é opcional em ambos os lados). No exemplo da Figura 7, tem-se as três situações representadas. Na primeira situação, Funcionario e Endereco teriam a mesma chave primária (matricula_funcionario) e puderam ser

unificadas. Na segunda situação, as chaves primárias são diferentes e optou-se por colocar a chave estrangeira na tabela Departamento, uma vez que este possui obrigatoriamente um chefe, associado ao fato de que nem todo funcionário ocupa um cargo de chefia. Na terceira e última situação, novamente as chaves são diferentes mas não existe obrigatoriedade em nenhum dos lados do relacionamento, não determinando assim o lado da chave estrangeira. Esta situação pode ser resolvida com uma nova tabela, como a tabela Chefia do exemplo.

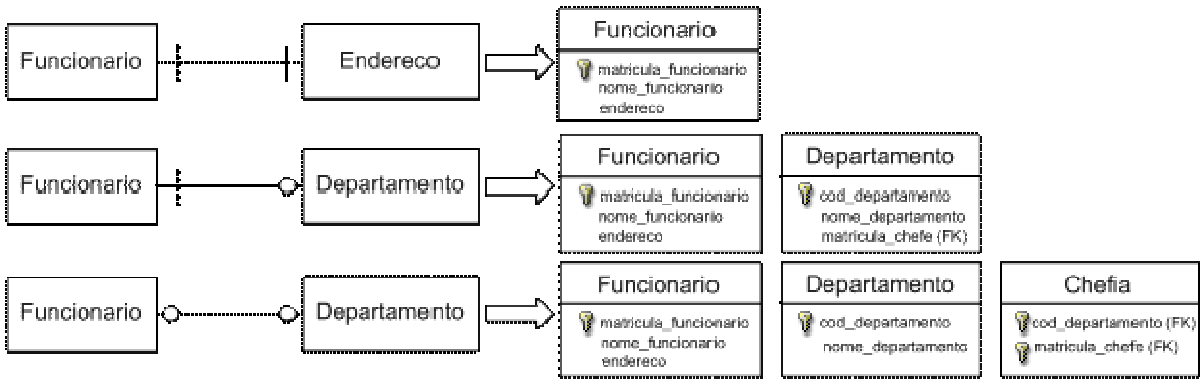

Figura 7. Mapeamento de relacionamentos 1:1.

29. Em relacionamentos 1:1, a chave pode ser migrada para qualquer entidade envolvida, contudo, é bom observar os seguintes critérios:

· havendo clara necessidade de recuperação de informação para um dos lados, migra-se o atributo para o lado da recuperação;

· Priorize colocar a chave estrangeira referenciando a obrigatoriedade do relacionamento, se existir.

30. Se houver necessidade do armazenamento de histórico, pode ser adotada como solução a criação de uma nova tabela que mapeará todos os elementos associados. Por exemplo, considere o modelo apresentado na Figura 8. Nele temos o relacionamento de 1:1 entre funcionário e departamento. Optou-se pela criação de uma terceira tabela que mapeará todos os funcionários que foram gerentes dos departamentos da empresa. Perceba a inclusão dos atributos data_inicio e data_fim, representando o período que determinado funcionário chefiou um departamento. Além disso, o atributo data_inicio foi colocado como parte da chave primária para permitir que um funcionário chefie o mesmo departamento em períodos distintos.

Figura 8. Exemplo de armazenamento de histórico.

31. Relacionamentos 1:N são mapeados de forma que a chave primária do lado “1” seja representada do lado “N” como chave estrangeira. Isso se deve ao fato do modelo relacional não permitir atributos multivalorados. Assim, a solução é representar o lado “1” do relacionamento. A Figura 6 exemplifica esta situação.

32. Relacionamentos N:N devem virar novas relações, associadas através de dois relacionamentos 1:n. Isso se deve ao fato de, no modelo relacional, não serem permitidos atributos multivalorados. Neste caso, a entidade criada possui como chave primária a concatenação das chaves das duas entidades relacionadas, formando uma chave composta. No exemplo da Figura 9, supondo que as chaves primárias das entidades Funcionario e Departamento sejam matricula e cod_departamento, respectivamente, a chave da entidade resultante Lotacao seria a união das duas chaves.

Figura 9. Transformação de relacionamento n:n em dois relacionamentos 1:n.

33. Um conjunto de passos pode ser seguido para identificar a cardinalidade de um relacionamento entre duas entidades. Deve-se ter em mente que a análise de todo relacionamento entre entidades é bidirecional. Assim, deve-se analisar como é determinada a cardinalidade existente entre, por exemplo, as entidades Cliente e Pedido (ver Figura 10).

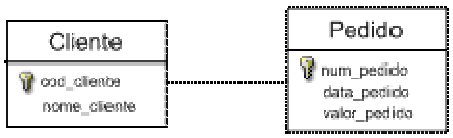

Figura 10. Entidades Cliente e Pedido.

· Passo 1: o primeiro passo será determinar a cardinalidade existente entre Cliente e Pedido. Considerando a navegabilidade no sentido Cliente à Pedido, realiza-se a seguinte pergunta: Um cliente pode fazer quantos pedidos? A resposta é: 1 cliente pode realizar vários pedidos. Então, a cardinalidade existente será de 1:N, pois a palavra “vários” em modelagem relacional é identificada pela letra N (ver Figura 11).

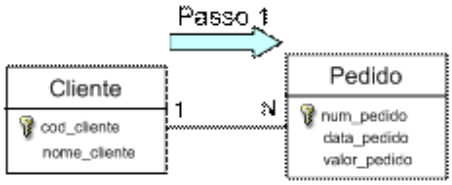

Figura 11. Entidades Cliente e Pedido – Passo 1.

· Passo 2: para realizar o segundo passo é necessário inverter a pergunta. Neste caso, determina-se a cardinalidade existente entre Pedido e Cliente. Percebe-se que se inverte a navegabilidade e realiza-se a seguinte pergunta: um Pedido pode possuir quantos clientes? A resposta é: 1 pedido pode possuir 1 cliente. Dessa forma, a cardinalidade existente será 1:1 (ver Figura 12).

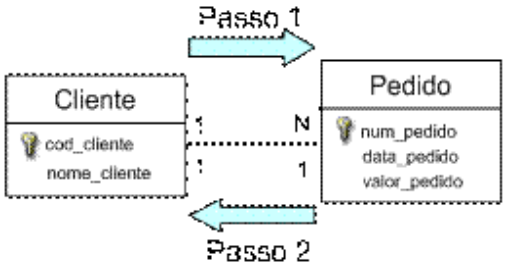

Figura 12. Entidades Cliente e Pedido – Passo 2.

· Passo 3: agora se tem em mãos a cardinalidade completa, ou seja, nas duas direções. O próximo passo é identificar a cardinalidade máxima para cada entidade definindo assim o relacionamento entre elas. Na posição vertical (identificada pelas elipses na Figura 13), tem-se a cardinalidade encontrada para cada entidade. No caso de Cliente, a maior cardinalidade será 1 e para Pedido será N.

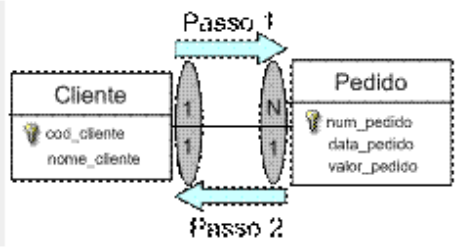

Figura 13. Entidades Cliente e Pedido – Passo 3.

· Passo 4: por fim, na Figura 14 exibe-se o relacionamento final existente entre as duas entidades. Aqui temos uma dica importante: após a implementação do modelo físico, o atributo cod_cliente será “duplicado” na tabela Pedido, garantindo assim o relacionamento entre as duas entidades. Isso quer dizer que um cliente não poderá ser eliminado enquanto seus respectivos pedidos também não forem. Essa é a terminologia padrão, porém pode ser ajustada a cada caso.

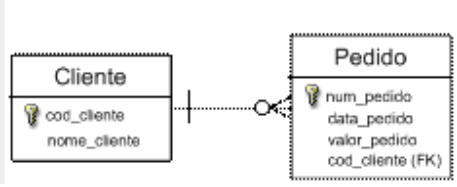

Figura 14. Entidades Cliente e Pedido – Final.

34. Relacionamentos do tipo Generalização/Especialização possuem, no modelo lógico, a mesma chave primária em todas as entidades participantes da hierarquia. A Figura 15 representa uma hierarquia de funcionários, onde todas as entidades possuem a mesma chave primária (matricula_funcionario). É comum que a entidade mais genérica (Funcionario) tenha um atributo que represente o tipo dos funcionários para facilitar que a hierarquia seja percorrida.

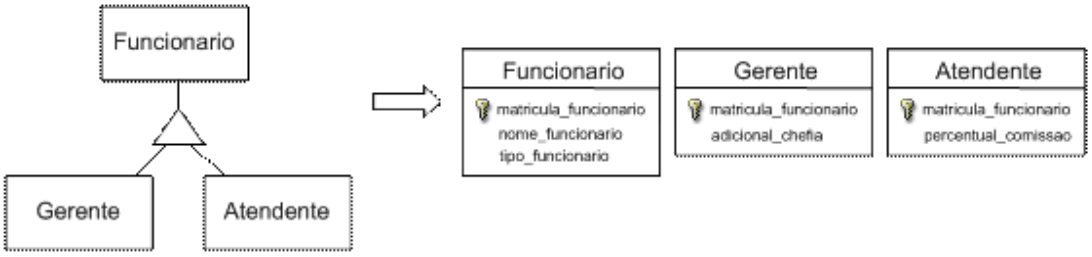

Figura 15. Hierarquia de entidades.

35. Generalização/Especialização podem ser mapeadas em uma única tabela, em uma tabela para cada especialização ou uma tabela para cada entidade envolvida, dependendo da situação. A Figura 16 apresenta as situações de mapeamento. Na situação 1, todos os atributos da hierarquia foram agrupados numa única tabela, fazendo com que atributos fiquem eventualmente vazios em função do tipo do funcionário. A situação 2 representa apenas as especializações, fazendo com que os atributos da generalização sejam repetidos. A situação 3 apresenta uma tabela para cada entidade da hierarquia, eliminando atributos repetidos, mas fazendo com que o acesso a um funcionário tenha que agrupar dados de mais de uma tabela.

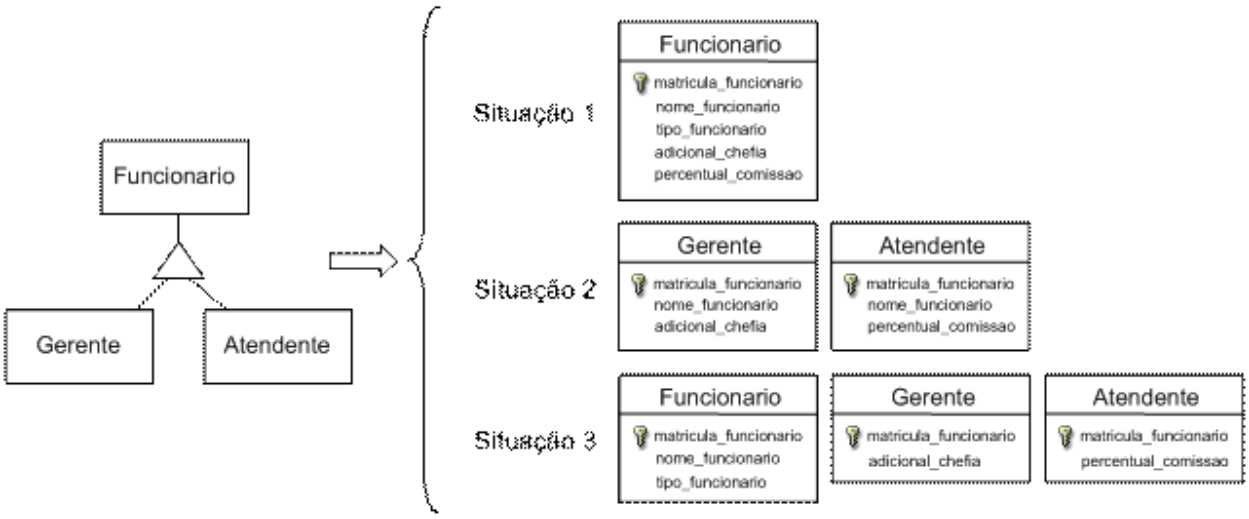

Figura 16. Generalização / Especialização.

36. Relacionamentos com atributos devem virar novas relações. Isso em função de não existir o conceito de relacionamento no projeto lógico. Assim, os atributos devem ser transferidos para uma relação. A Figura 3 exemplifica esta situação.

37. A chave primária de uma entidade fraca deve ser formada pela chave primária da entidade forte e mais algum outro campo que diferencie os registros. A Figura 17 apresenta a entidade fraca ItemPedido, cuja chave primária é formada pela chave de sua entidade forte (num_pedido) acrescida de um atributo próprio (num_item_pedido).

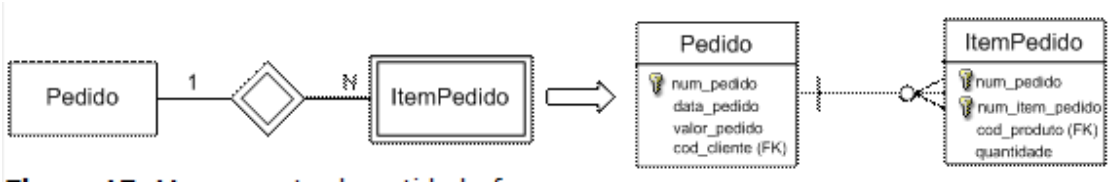

Figura 17. Mapeamento de entidade fraca.

38. Entidades fracas indicam que, ao eliminar a entidade forte correspondente, devem ser eliminadas todas as ocorrências das entidades fracas, uma vez que a chave primária da entidade forte compõe a chave primária da entidade fraca, e não pode virar nulo.

39. Atenção para relacionamentos com grau maior do que 2. Estes relacionamentos geram novas tabelas cujas chaves primárias referem-se às chaves das relações participantes ou cria-se uma nova chave primária conforme exemplo da Figura 18. Neste exemplo, o relacionamento N:N entre Pedido e Produto deverá ser decomposto em outros dois relacionamentos 1:N.

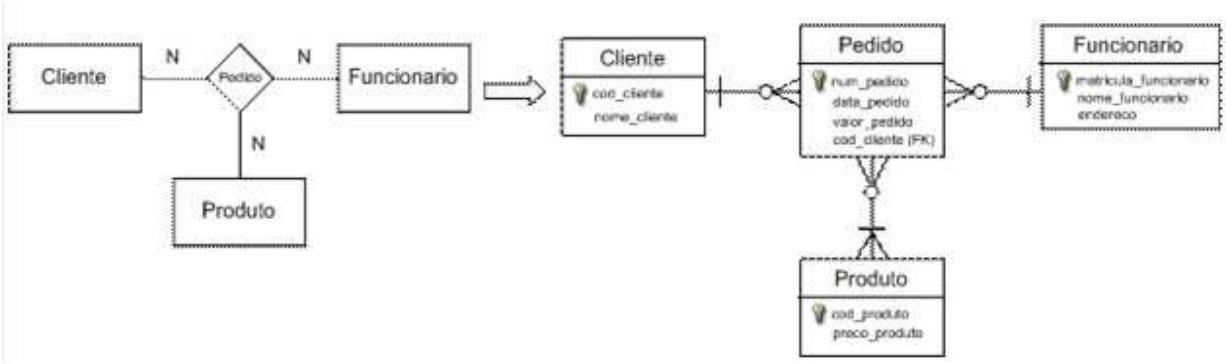

Figura 18. Mapeamento de relacionamento ternário.

40. Agregações normalmente originam novas tabelas. A Figura 19 representa uma agregação entre produtos e promoções, onde cada item de um pedido possui um produto em uma determinada promoção.

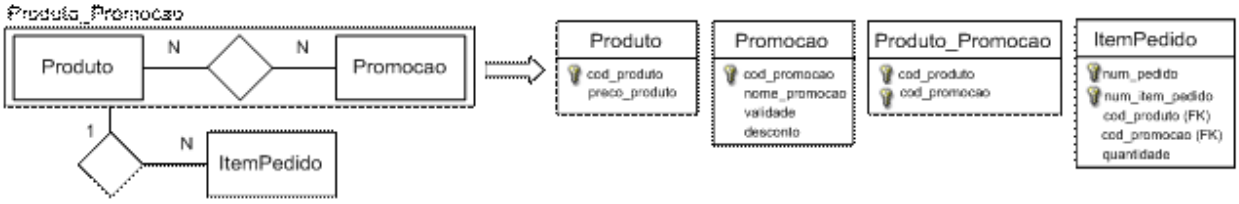

Figura 19. Mapeamento de agregação.

41. Auto-relacionamentos do tipo 1:N geram um atributo de ligação na própria tabela, como demonstrado na Figura 20. Esta representa que um cliente pode indicar vários clientes mas, por outro lado, um cliente só pode ser indicado por um outro. Sendo do tipo n:n, geram novas tabelas.

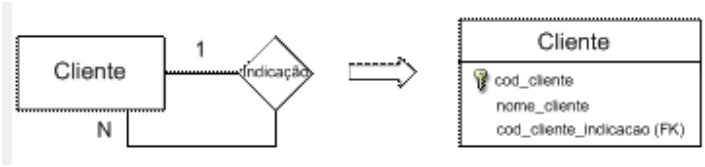

Figura 20. Mapeamento de auto-relacionamento.

Normalização

42. Faça sempre normalização dos dados. É uma forma de verificar sua qualidade, diminuindo redundâncias e incoerências no modelo de dados. O processo de normalização aplica uma série de regras sobre as tabelas de um modelo para verificar se estas estão corretamente projetadas. Embora existam seis formas normais, ou regras de normalização (são cinco Formas Normais e a Forma Normal de Boyce-Codd), normalmente utilizam-se apenas as três primeiras formas normais.

43. Uma das preocupações da Primeira Forma Normal refere-se a eliminar atributos compostos. Assim, atributos como Endereço devem ser decompostos em Logradouro, Número, Complemento, Bairro, Cidade, Estado, CEP, exemplificado na Figura 21.

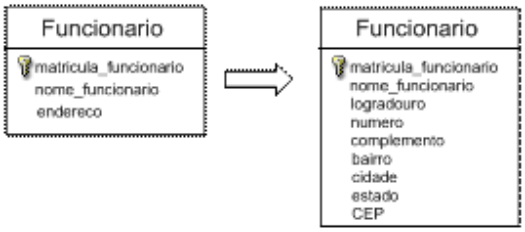

Figura 21. Análise da primeira forma normal – atributo composto.

44. Se a Primeira Forma Normal não for obedecida, corre-se o risco de perda de informações. Por exemplo, se o atributo Endereço não for decomposto, como saber se alguém mora na cidade do Rio de Janeiro ou na Rua Rio de Janeiro em Belo Horizonte?

45. A Primeira Forma Normal também se refere a eliminar atributos multivalorados. A Figura 22 exemplifica a normalização do atributo Telefones, que deve ser decomposto em diversos telefones, como comercial, celular e residencial (situação 1), ou então virar uma nova tabela (situação 2).

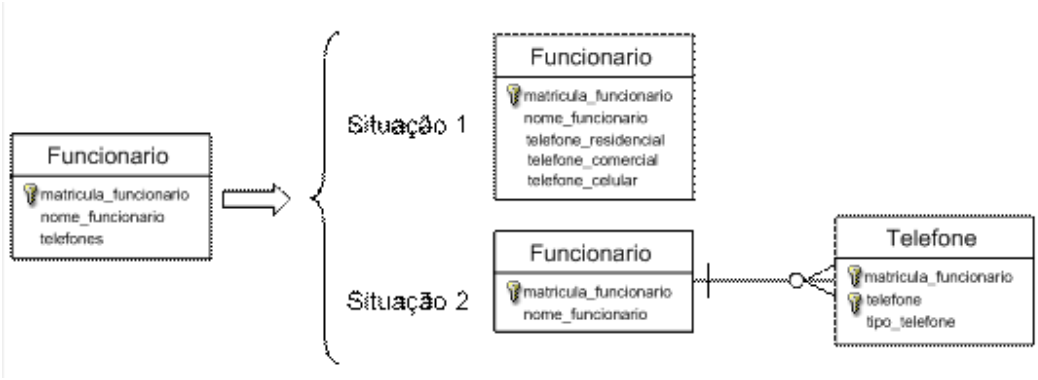

Figura 22. Análise da primeira forma normal – atributo multivalorado.

46. Na Primeira Forma normal deve-se observar se todas as relações possuem sua chave primária definida. Isto é importante pois as formas normais seguintes baseiam-se na definição da chave primária.

47. Antes de realizar a análise da Segunda Forma Normal, deve-se garantir que todas as relações estejam na Primeira Forma Normal.

48. A Segunda Forma Normal determina que atributos não chave devem ser funcionalmente dependentes apenas da chave primária. Ou seja, deve-se analisar se todo atributo que não é chave primária depende totalmente dela. Assim, não é necessário se preocupar com esta forma normal para tabelas que tenham chaves primárias simples, com apenas um atributo.

49. Considere a relação entre Pedido e ItemPedido conforme demonstrada inicialmente na Figura 23. Imagine que o atributo data_pedido tivesse sido colocado na tabela ItemPedido. Através da análise da Segunda Forma Normal nesta tabela (que possui chave primária composta), pode-se verificar que a data_pedido depende apenas de parte da chave primária, o num_pedido, e não da chave primária completa, composta por num_pedido e num_item_pedido. Sem esta análise, a data do pedido seria repetida para todos os itens de um mesmo pedido. A verificação da Segunda Forma Normal, neste exemplo, determina que o atributo data_pedido deva ser transferido para outra entidade.

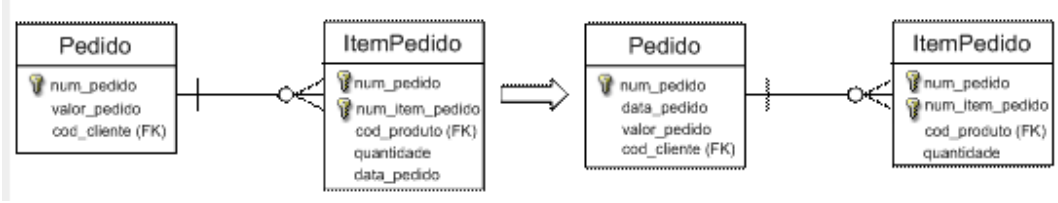

Figura 23. Análise da segunda forma normal.

50. Outras implicações no exemplo da dica anterior seriam relativas a anomalias de inclusão, alteração e exclusão. Para anomalia de inclusão, pode-se verificar que não é possível inserir a data do pedido enquanto não se registra um item do pedido. Para anomalia de exclusão, se forem excluídos todos os itens do pedido, perde-se a data do mesmo. Para anomalia de alteração, se a data do pedido precisar mudar, deve-se mudar a mesma em todos os seus itens.

51. Antes de proceder com a análise da Terceira Forma Normal, deve-se garantir que todas as relações estejam na Segunda Forma Normal.

52. A Terceira Forma Normal refere-se a atributos não chave mutuamente independentes, ou seja, que não dependam um do outro. Desta forma, é verificado se existe dependência funcional entre atributos não chave.

53. Considere a relação Funcionario conforme apresentada inicialmente na Figura 24. Pode-se verificar que esta relação está na Primeira e Segunda Formas Normais. Entretanto, existe um atributo não chave (nome_departamento) que depende de outro atributo não chave (cod_departamento). Este é o tipo de problema de que trata a Terceira Forma Normal. Para resolvê-lo, cria-se uma nova relação, no caso Departamento, contendo os atributos relacionados e mantém-se na relação original apenas aquele que determina o outro como chave estrangeira.

Figura 24. Análise da terceira forma normal.

54. Ao não se obedecer a Terceira Forma Normal, corre-se os mesmos riscos apresentados na Segunda Forma Normal. Por exemplo, não se pode inserir um departamento antes que se tenha um funcionário alocado nele. A eliminação de todos

os funcionários de um departamento acarreta na eliminação das informações do próprio departamento. No caso de um departamento mudar de nome, esta modificação deve ser tratada em todos os funcionários.

Desnormalização

55. O processo inverso à normalização é chamado de desnormalização. Nesse processo, uma ou mais entidades do modelo conceitual são aglutinadas em uma única tabela de um esquema. A princípio, a única razão para a aplicação da desnormalização é a de eliminar o custo das junções em operações de seleção sobre as tabelas envolvidas. Entretanto, uma análise superficial dessa frase pode levar o leitor a concluir que a desnormalização sempre aumenta o desempenho no processamento de consultas de seleção. O raciocínio (errôneo) para essa conclusão seria o seguinte: "se a quantidade de tabelas é maior em um esquema relacional normalizado, então haverá um maior número de operações de junção para a obtenção do resultado das consultas nesse esquema do que em um esquema desnormalizado correspondente (operações de junção são sabidamente bastante custosas do ponto de vista computacional). Conseqüentemente, o custo da execução de seleções em um esquema normalizado é sempre maior do que o custo sobre o esquema desnormalizado correspondente".

56. Considerando as junções, pode-se constatar que a sobrecarga de processamento necessária para manter a integridade dos dados pode não compensar o ganho de desempenho obtido com a desnormalização. De fato, a manutenção da integridade pode necessitar das mesmas operações de junção que a desnormalização se propunha a eliminar! Para exemplificar, considere uma versão desnormalizada da tabela Pedido contendo as colunas das tabelas ItemPedido e Produto (ver Figura 25). Para obter informações sobre pedidos, itens de pedidos e produtos, a utilização dessa tabela desnormalizada realmente produz uma execução mais eficiente. Afinal de contas, todos os dados necessários estão armazenados em uma única tabela, o que normalmente leva o SGBD a posicionar essas informações em blocos de disco contíguos. No entanto, o que acontece quando um nome de produto é atualizado? Uma das operações que devem ser realizadas para garantir que os dados permaneçam consistentes é atualizar todos os registros em que o produto em questão está contemplado. Deixa-se como exercício para o leitor verificar que há também problemas quando da exclusão e inclusão de registros nessa tabela. Todos esses problemas provenientes da sua desnormalização!

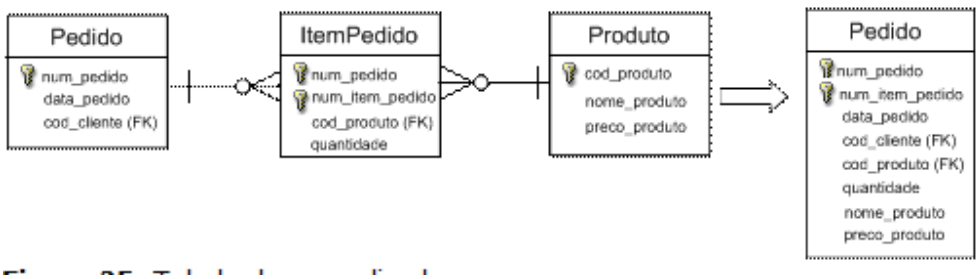

Figura 25. Tabela desnormalizada.

57. Um eventual benefício obtido pela desnormalização (aumento do desempenho em uma determinada consulta de seleção) tem seu preço: uma tabela desnormalizada fica vulnerável ao surgimento de anomalias quando manipulações (atualizações, inclusões ou inserções) são realizadas sobre ela, e a integridade dos dados fica ameaçada.

58. A sobrecarga necessária para manutenção da integridade dos dados não é o único fator a considerar quando o desenvolvedor estiver pensando em desnormalizar um esquema. Há um outro aspecto menos óbvio. Considere novamente os dados da tabela Pedido (ver Figura 25). Essa tabela armazena dados sobre três conceitos distintos: pedidos, itens de pedidos e produtos. O que acontece quando aplicações de bancos de dados precisam obter acesso a esses dados separadamente? Por exemplo, digamos que o departamento de logística precisa fazer um levantamento sobre os produtos

em estoque. Provavelmente, essa tarefa envolveria uma consulta sobre a tabela Produto para resgatar somente os dados relativos à quantidade em estoque por produto. Como esses dados estão em uma tabela que possui diversas outras colunas não relacionadas a produtos, a quantidade de informações sobre produtos por bloco de disco será menor do que se houvesse uma tabela armazenando somente dados sobre produtos. Conseqüentemente, o SGBD levará mais tempo para resgatar os dados necessários para montar o relatório de estoque ao utilizar a tabela desnormalizada. Perceba que esse ponto acaba indo de encontro ao citado anteriormente neste artigo, ou seja, um dos benefícios da desnormalização (o não uso de junções) acaba sendo prejudicado em outras situações.

59. De uma forma geral, se for tomada a decisão de aglutinar dados de duas ou mais tabelas em uma única tabela (ou seja, desnormalizar), as aplicações que necessitam ter acesso aos dados, que do contrário estariam em uma tabela separada, terão agora que ler desnecessariamente outras informações. E a leitura dessas outras informações desnecessárias aumenta o tempo de processamento das consultas de seleção.

60. Vamos agora a um exemplo em que o uso da desnormalização pode ser benéfico. Imagine uma situação em que tenhamos as tabelas Produto e GrupoProduto (ver Figura 26). A tabela GrupoProduto possui uma classificação fixa para categorizar os diferentes produtos (ou seja, o nome dos grupos não muda). Neste caso, poderíamos colocar o campo nome_grupo na tabela Produto para facilitar, por exemplo, a geração de relatórios.

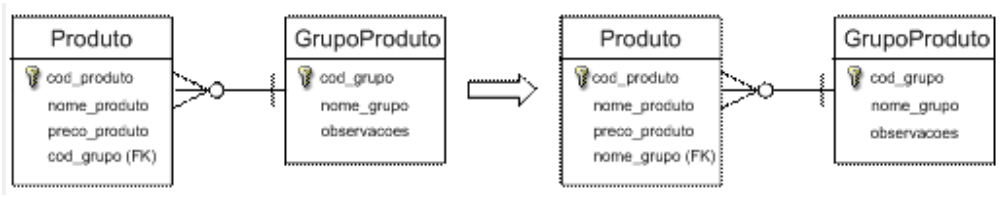

Figura 26. Exemplo de desnormalização.

61. Imagine uma situação em que você precise constantemente retornar um valor de um campo calculado. Pode ser interessante criar um campo valor_pedido do pedido na tabela Pedido para que, toda vez que precisar calcular o valor do pedido não seja necessário percorrer seus itens e depois, os produtos, para chegar neste valor (ver Figura 27). Entretanto, temos também uma desvantagem: a cada mudança no pedido, este campo precisa ser recalculado e armazenado. Teoricamente, campos deste tipo (calculados), não precisariam ser armazenados.

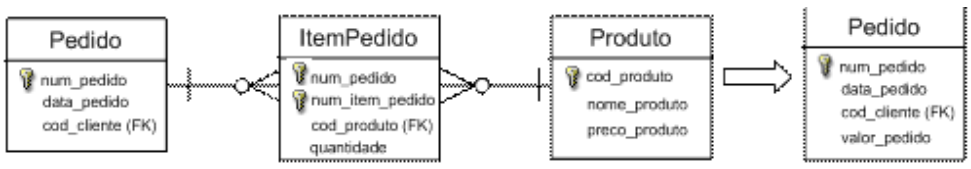

Figura 27. Exemplo de desnormalização.

62. Outra situação interessante de uso da desnormalização é a criação de histórico. No exemplo da Figura 28 pode-se perceber que se repete o valor do produto na tabela item_pedido. Isso pode ser interessante, pois, caso o valor do produto

mude (na tabela Produto), mantêm-se o histórico do valor praticado naquele item_pedido. Vale ressaltar que isso contraria a Terceira Forma Normal, pois o valor do produto depende do código mas, em função do histórico, é justificável.

Figura 28. Exemplo de desnormalização.

63. Uma outra situação de uso para desnormalização é a eliminação de uma tabela quando pudermos pré-determinar os tipos de valores que ela contém, juntando na tabela original. No exemplo da Figura 29, como os tipos de telefones são conhecidos, justifica-se eliminar a tabela Telefone e agrupar estes dados em Funcionário (neste caso estamos considerando apenas um telefone por tipo).

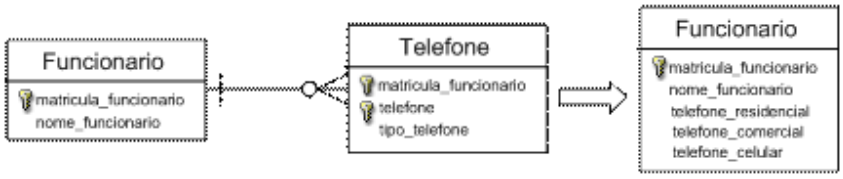

Figura 29. Exemplo de desnormalização.

Modelagem de dados: projeto físico

64. O projeto físico se preocupa com questões de desempenho, criação de índices, particionamento, paralelismo e tuning, dentre outros. Assim, diversas características do projeto físico estão associadas a um SGBD específico.

65. A criação de índices para chaves estrangeiras tendem a melhorar o desempenho de consultas complexas ou que envolvam muitas tabelas ou ainda tabelas com muitos registros.

66. Verificar a necessidade de chaves artificiais (Surrogate Key): em alguns casos depara-se com certas tabelas que não possuem uma chave primária natural, o que fará com que haja a necessidade de se criar uma nova coluna para ser a chave primária. Neste caso, temos a criação de um novo campo único que represente sua chave, como um seqüencial, por exemplo.

67. Validação do modelo: antes de gerar os scripts de criação do banco de dados, é fundamental verificar o modelo para identificar se este responde aos resultados esperados.

68. Atente para as configurações necessárias a serem realizadas nas ferramentas CASE antes da geração do script. Muitas delas trazem opções específicas para cada SGBD.

69. Não utilize muitos atributos nas chaves nas tabelas: apesar de muitas vezes isso acontecer no modelo lógico, quando se implementa no banco de dados um modelo com muitos atributos na chave complica-se desnecessariamente a programação. A sugestão é a criação de chaves alternativas que facilitam o acesso aos dados.

Fragmentação de relações

70. Para desenvolver um projeto de banco de dados adequado, o projetista, dentre as muitas decisões que deve tomar para o sucesso ou fracasso do projeto, deve decidir também se os dados serão distribuídos pelo local de uso ou serão centralizados e acessados via rede pelos usuários. Essa decisão envolve custos de infra-estrutura, perda ou melhoria de desempenho no acesso aos dados, interferindo também na disponibilidade imediata ou posterior desses dados.

Algumas razões para não realizar a fragmentação

71. Quando não houver necessidade de deixar os dados próximos aos usuários: neste caso, não deve haver a necessidade de grandes operações de manipulação de dados por parte dos usuários;

72. Quando a fragmentação gerar muita duplicidade trazendo problemas de consistências: muitas vezes ao realizarmos a fragmentação, precisamos realizar a duplicação de alguns registros. Quando a política de replicação e/ou sincronização for um problema, evite fragmentar sua base de dados;

73. Quando houver restrições tecnológicas que impeçam a fragmentação;

74. Quando houver dificuldade em manter a distribuição otimizada dentro de um cenário de mudança de aplicações: este tópico também diz respeito aos processos de replicação e sincronização. Ambos não são, na maioria das vezes, triviais de serem realizados, portanto, tenha isso em mente.

Algumas razões para fragmentar

75. Quando os dados precisam estar próximos dos seus usuários: isso ocorre quando há necessidade de grandes operações de manipulação de dados por parte dos usuários. Se os dados não estiverem localizados no ambiente operacional, estas operações se tornarão bastante custosas e, a depender do caso, bastante lentas;

76. Quando a decomposição de uma relação em fragmentos permitir um maior número de transações em execução concorrente.

77. Dessa forma, devemos considerar a realização da fragmentação quando a distribuição dos dados traz vantagens considerando, principalmente a localização dos dados e o desempenho das operações de manipulação de dados. Além destes tópicos, pode-se ainda ter a necessidade de realizar a fragmentação por questões estratégicas da organização. Por exemplo, parte dos dados de uma base podem ser fundamentais para o sucesso do negócio, ao invés de mantê-los juntos aos outros dados, podemos fragmentar a base de forma que estes dados mais sensíveis sejam colocados em outro servidor com políticas mais severas de segurança.

Técnicas básicas de fragmentação

Fragmentação vertical

78. A fragmentação vertical ocorre quando colunas da mesma tabela são separadas em locais diferentes. Conforme pode ser visto no exemplo da Tabela 1, todas as colunas estão centralizadas em uma única tabela e nas Tabelas 2 e 3 as colunas estão fragmentadas de forma vertical. Essa fragmentação será vantajosa quando os aplicativos dos diversos usuários puderem atuar de forma separada em cada fragmento minimizando desta forma o tempo de processamento.

Tabela 1. Relação Cliente sem fragmentação.

Número	Data abertura	Nome Cliente	Endereço	Filial	Limite
1.111.111	12/12/1995	Paulo de Castro Silva	Rua das Flores, 87	CT	5.000,00
2.222.222	15/07/2000	Roberto Carlos Silveira	Rua Mainacá, 1980	BH	3.000,00
3.333.333	21/12/2001	Alberto de Moura	Rua Fernandópolis, 789	BH	4.000,00
4.444.444	12/08/2001	Cláudio Bassi	Rua dos Pardais, 33	SP	6.000,00
5.555.555	06/09/2001	Ronaldo Servic	Av. Carlos Alves, 899	SP	4.000,00
6.666.666	14/09/2002	Carla dos Santos	Trav. Paulo Emidio, 45	SP	3.000,00

Tabela 2. Fragmento vertical da Relação Cliente sem o atributo Limite.

Número	Data abertura	Nome Cliente	Endereço	Filial
1.111.111	12/12/1995	Paulo de Castro Silva	Rua das Flores, 87	CT
2.222.222	15/07/2000	Roberto Carlos Silveira	Rua Mainacá, 1980	BH
3.333.333	21/12/2001	Alberto de Moura	Rua Fernandópolis, 789	BH
4.444.444	12/08/2001	Cláudio Bassi	Rua dos Pardais, 33	SP
5.555.555	06/09/2001	Ronaldo Servic	Av. Carlos Alves, 899	SP
6.666.666	14/09/2002	Carla dos Santos	Trav. Paulo Emidio, 45	SP

Tabela 3. Fragmento vertical da Relação Cliente contendo apenas a chave e o limite.

Número	Limites
1.111.111	5.000,00
2.222.222	3.000,00
3.333.333	4.000,00
4.444.444	6.000,00
5.555.555	4.000,00
6.666.666	3.000,00

79. É importante ressaltar que se uma relação for decomposta verticalmente em outras, então qualquer registro existente na primeira relação também deve ser encontrado nas demais, para que haja a correta reconstrução do registro.

Fragmentação horizontal

80. A fragmentação horizontal ocorre quando registros de uma mesma tabela são separados em fragmentos diferentes. Conforme pode ser visto no exemplo mais adiante, cada filial armazenará os dados sobre os seus clientes.

81. A questão que se deve ter em mente ao optar por uma fragmentação horizontal é: as consultas têm visões locais dos dados? Em caso afirmativo, com certeza haverá melhora de desempenho ao se evitar ler dados de clientes de outras filiais e evitar acesso remoto desnecessário. Além disso, haverá aumento no número de transações concorrentes. A tabela base para nosso exemplo é novamente a Tabela 1 sem fragmentação. Nas Tabelas 4, 5 e 6 temos os fragmentos horizontais da tabela Cliente onde os clientes são separados por filial.

Tabela 4. Fragmento horizontal da tabela Cliente contendo somente dados da Filial Belo Horizonte.

Número	Data abertura	Nome Cliente	Endereço	Filial	Limite
2.222.222	15/07/2000	Roberto Carlos Silveira	Rua Mainacá, 1980	BH	3.000,00
3.333.333	21/12/2001	Alberto de Moura	Rua Fernandópolis, 789	BH	4.000,00

Tabela 5. Fragmento da tabela Cliente contendo somente da Filial Curitiba.

Número	Data abertura	Nome Cliente	Endereço	Filial	Limite
1.111.111	12/12/1995	Paulo de Castro Silva	Rua das Flores, 87	CT	5.000,00

Tabela 6. Fragmento da tabela Cliente contendo somente da Filial São Paulo.

Número	Data abertura	Nome Cliente	Endereço	Filial	Limite
4.444.444	12/08/2001	Cláudio Bassi	Rua dos Pardais, 33	SP	6.000,00
5.555.555	06/09/2001	Ronaldo Servic	Av. Carlos Alves, 899	SP	4.000,00
6.666.666	14/09/2002	Carla dos Santos	Trav. Paulo Emidio, 45	SP	3.000,00

Para que a fragmentação ocorra de forma adequada, alguns critérios importantes devem ser considerados:

82. No caso de fragmentação horizontal, se um registro da tabela está em um fragmento, não poderá estar também em outro fragmento.

83. Se houver uma tabela que dependa diretamente de outra, os registros relacionados devem ficar no mesmo local.

84. Para efetuar uma fragmentação horizontal, devemos considerar: a cardinalidade das tabelas; os critérios de seleções por aplicação; a quantidade de registros recuperados por aplicação e por local; a freqüência de busca dos registros por aplicação e por local.

85. Deve-se considerar para cada fragmento a cardinalidade dos fragmentos e o tamanho de cada registro por fragmento;

86. Deve-se considerar para cada aplicação os fragmentos acessados, os critérios de seleção de registros e quantidade de registros selecionados por acesso de leitura, atualizações em registros, freqüência de execução por local de ativação e tempo máximo de resposta esperado;

87. Deve-se considerar para cada estação servidora, a capacidade de processamento e grau de utilização e também a capacidade de armazenamento e grau de utilização;

88. Deve-se considerar para a rede de comunicação, a capacidade da banda e grau de utilização da rede.

89. Razões de restrições tecnológicas também exigirão que sejam feitas escolhas de atualizações que melhor se adaptem à infra-estrutura de cada organização. Essa escolha pode envolver atualizações assíncronas que são aquelas atualizações que ocorrem em horários específicos para não prejudicar a performance do banco de dados, ou síncronas onde as atualizações ocorrem em vários locais simultaneamente. Essa escolha, como já foi dito, trará conseqüências sobre a performance do banco de dados e por isso deverá ser extremamente criteriosa. A seguir são apresentadas algumas técnicas de atualização de bancos de dados distribuídos:

· Extratos: são cópias que nunca serão atualizadas. Úteis em situações que exigem apenas consultas. O extrato é feito com ajuda de comandos SQL. São utilizadas, por exemplo, em sistemas de apoio à decisão. Esses extratos podem ter ou não o registro do momento de sua criação, dependendo da necessidade desse tipo de informação.

· Extrato com substituição periódica: neste caso a cópia é substituída em intervalos de tempos pré-definidos. São utilizadas, por exemplo, em sistemas de suporte a decisão e sistemas distribuídos que utilizam dados atualizados com periodicidade bem definida.

· Réplicas com aplicação assíncrona de atualizações: as atualizações são feitas nas cópias e no banco de dados original. Periodicamente, os bancos de dados são re-sincronizados. Podem ser usadas em sistemas onde as alterações dos bancos de dados sejam feitos por inclusões em qualquer um dos locais.

· Réplicas com aplicação síncrona de atualizações: as atualizações são feitas nas cópias e no BD original no momento de sua ocorrência. Podem ser utilizadas em sistemas onde as alterações dos bancos de dados sejam pouco freqüentes e possam ser feitas em qualquer um dos locais.

Ferramentas de modelagem

90. Utilize uma ferramenta CASE para modelagem de dados. Caso contrário, o modelo tenderá a ficar desatualizado rapidamente, pois dificilmente as modificações que são necessárias no esquema de dados são refletidas para a documentação sem o auxílio de uma ferramenta apropriada.

91. Até pouco tempo atrás, a realização das atividades de modelagem era dificultada para aqueles que não possuíam recursos financeiros para comprar ferramentas de modelagem comerciais. Ficava assim difícil realizar atividades de modelagem. Com o advento da Internet e do software de código aberto essa realidade foi se transformando e hoje já é possível encontrar algumas destas ferramentas gratuitas para modelagem de dados. Um bom exemplo disso é a ferramenta DBDesigner 4 (ver Figura 30). Eis algumas das características que a tornam uma boa opção:

· Apresenta uma interface de fácil utilização;

· Atende aos SGBDs: MySQL, Oracle, SQL Server, SQLite, e todos outros que suportem acesso via ODBC;

· Possui engenharia reversa;

· Salva os arquivos em formato XML;

· Importa modelos gerados por algumas outras ferramentas;

- Gera relatórios em HTML;
- Suporte através do fórum disponível no site do DBDesigner;
- Realiza sincronia no banco de dados a partir das alterações no modelo;
- É multi-plataforma.

Para realizar o download, basta entrar no site .

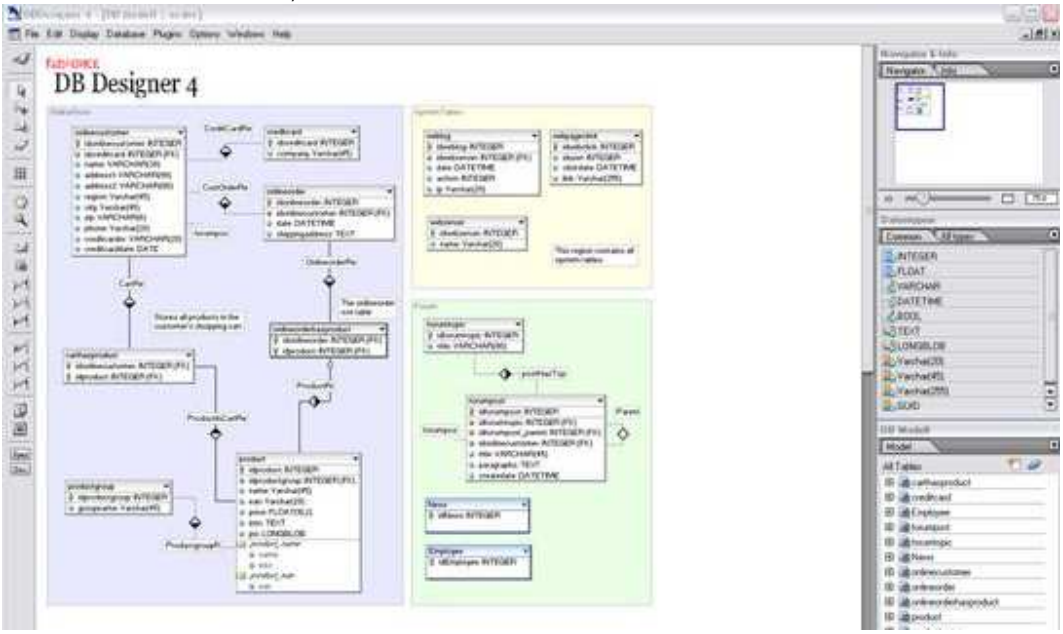

Figura 30. DBDesigner 4.