Banco de Dados 1: Fundamentos
Turma 2024A

Descrição do Curso

Neste curso abordaremos os conceitos básicos de banco de dados, incluindo a modelagem relacional e normalização. Tais conceitos serão essenciais para que você prossiga nos estudos e comece a desenvolver programas que utilizem a linguagem SQL. O curso utiliza como base os materiais produzidos para Rede e-Tec Brasil e outros materiais complementares. O curso é organizado em quatro módulos, totalizando 20 horas.

Gustavo Pires Bertaco

Junho 2024

Sumário

1 Conceitos fundamentais **2**
1.1 Introdução 2
1.2 História 4
1.3 Usuários (atores de um banco de dados) 4
1.4 Níveis de abstração 5
1.5 Sistemas Gerenciadores de Bancos de Dados - SGBD 6
1.6 Vídeo complementar: Conceitos fundamentais sobre bancos de dados 10
1.7 Vídeo complementar 2: Conceitos fundamentais sobre bancos de dados 10

2 Modelos e modelagem **10**
2.1 Modelos de dados 10
2.1.1 Modelo hierárquico 11
2.1.2 Modelo em rede 11
2.1.3 Modelo de dados orientado a objeto 12
2.2 Modelagem de dados 13
2.2.1 Relacionamentos 16
2.2.2 Link: Software DIAURL 19

3 Modelo de dados relacional **19**
3.1 Introdução 19
3.2 Conceitos 19
3.3 Atributos-chaves 21
3.4 Integridade referencial 24
3.5 Mapeamento 24

4 Normalização de dados **26**
4.1 Introdução 26
4.2 Formas normais 27
4.3 Dependência funcional 28
4.4 Normalização de dados (nota fiscal) 28
4.5 Normalização de dados (formulário remanejamento) 31
4.6 Aplicação da 1ª Forma Normal - 1FN 32
4.7 Aplicação da 2ª Forma Normal - 2FN 33

5 Referencias **34**

1 Conceitos fundamentais

1.1 Introdução

Todos nós, em nosso dia a dia, temos a necessidade de armazenar e recuperar dados. Cadernos de endereços, listas de telefones, dados financeiros, receitas, enfim, estamos sempre lançando mão de memórias auxiliares. O ideal seria que estas memórias pudessem ser seguras, confiáveis e estivessem disponíveis quando precisássemos dela. Um dado guardado, mas que não sabemos como recuperar é praticamente tão inútil quanto se não existisse. Assim, temos uma grande necessidade de armazenar informações que não se encontram isoladas, como por exemplo, as fichas de matricula de um aluno que contém informações diversas sobre ele, conforme a figura abaixo:

INSTITUTO FEDERAL
Rio Grande do Sul

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DO RIO GRANDE DO SUL

FICHA INDIVIDUAL DO ALUNO

NOME:	
ENDEREÇO:	
RG:	
PRONTUÁRIO:	
CURSO:	
ANO DE INGRESSO:	

Figura 1: Exemplo de Ficha de Matrícula

Além de uma forma adequada para definir o armazenamento destas informações, os usuários desejam realizar operações sobre esta coleção de dados, tais como: adicionar novos dados, consultar um determinado subconjunto de dados, atualizar ou modificar a estrutura dos dados e eliminar informações desnecessárias ou que já perderam a validade por um motivo ou outro, de desatualização ou desuso.

Uma solução para este problema foi apresentada com o advento da tecnologia em Bancos de Dados (BD, Database em Inglês). Para que você comece a dominar o tema em estudo apresentamos várias definições, assim um banco de dados é uma coleção de dados relacionados. Os dados são fatos que podem ser gravados e possuem um significado implícito. Assim, considere o exemplo anterior da ficha com dados de um aluno ou uma lista telefônica. Esses dados podem ser armazenados em uma ficha, uma agenda ou em um computador por meio de aplicativos como a Microsoft Excel ou Access. Essas informações têm um significado implícito, formando um banco de dados. Algumas definições mais precisas de Banco de Dados podem ser conhecidas abaixo:

- Um Banco de Dados é uma coleção logicamente coerente de dados com um determinado significado inerente. Isto significa que um conjunto aleatório

de dados não pode ser considerada um Banco de Dados;

- Um Banco de Dados é projetado, construído e composto por um conjunto de dados para um **propósito específico** (como no nosso exemplo acima para armazenar um conjunto de informações de alunos). Existe um grupo de usuários ou algumas aplicações pré-concebidas onde estes dados serão utilizados;

- Um Banco de Dados representa aspectos de uma parte restrita do mundo real, denominado de minimundo. Alterações que ocorrem no minimundo são refletidas em todo o Banco de Dados.

Um banco de dados computadorizado é mantido por um grupo de aplicativos chamado de Sistema de Gerenciamento de Bancos de Dados (SGBD), que é uma **coleção de programas** que permite ao usuário definir, construir e manipular um Banco de Dados para as mais diversas aplicações. O objetivo principal de um sistema de banco de dados é possibilitar um ambiente que seja adequado e eficiente para uso na recuperação e armazenamento de informações.

Figura 2: Sistema Gerenciador de Banco de Dados (SGBD)

Banco de Dados e seu software são juntos denominados de Sistema de Bancos de Dados (SBD).

Imagine um armário de aço, com várias gavetas, em cada gaveta contém alguma informação (como a ficha do aluno) que estão agrupadas de acordo com seu tipo. O armário no caso é forma de gerenciamento dos dados ali contidos, lá podemos: inserir, excluir, selecionar ou alterar algum documento que

ali contenha. Neste primeiro momento podemos pensar que um banco de dados computacional consiste em "levar" os dados deste armário de aço para o computador, porém seguiremos algumas regras para que o armazenamento seja mais eficiente.

1.2 História

Os primeiros Sistemas de Bases de Dados foram lançados no mercado no final da década de 60 e eram conceitualmente muito simples, de acordo com as necessidades das aplicações da época. No inicio as empresas que impulsionaram este segmento foram a IBM, ORACLE e SYBASE. Assim, como outras tecnologias na computação, os fundamentos de bancos de dados relacionais nasceram na empresa IBM, nas décadas de 1960 e 1970, por meio de pesquisas de funções de automação de escritório. Foi durante um período da história nas quais as empresas verificaram que era necessário empregar muitas pessoas, além do alto custo para fazer trabalhos, tais como: armazenar e organizar seus arquivos. Por este motivo, eram importantes os esforços e investimentos em pesquisa para obter-se um meio mais barato e eficiente de armazenamento de dados.

Em 1970 a IBM publicou o primeiro trabalho sobre bancos de dados relacionais. Este trabalho tratava sobre o uso de cálculo e álgebra relacional para que usuários não técnicos pudessem manipular grande quantidade de informações.

Devido à natureza técnica e a relativa complicação matemática, o significado e proposição do trabalho não foram prontamente realizados. Assim, a IBM montou um grupo de pesquisa conhecido como System R.

O projeto do Sistema R tinha como proposta a criação de um sistema de banco de dados relacional o qual eventualmente se tornaria um produto. Os primeiros protótipos foram utilizados por várias organizações, como MIT Sloan School of Management. Novas versões foram testadas com empresas de aviação.

O Sistema R evoluiu para SQL/DS tornando-se o DB2 (Database 2, em inglês). O grupo do Sistema R criou a Structured Query Language (SQL) - Linguagem de Consulta Estruturada. Esta linguagem tornou-se um padrão na indústria para bancos de dados relacionais e hoje em dia é um padrão ISO (International Organization for Standardization).

Mesmo a IBM sendo a empresa que inventou o conceito original e o padrão SQL, ela não produziu o primeiro sistema comercial de banco de dados. Isto foi realizado pela Honeywell Information Systems Inc., tendo seu sistema lançado em junho de 1976. O sistema foi desenvolvido com base em muitos dos princípios que a IBM concebeu, mas foi modelado e implementado fora da IBM.

1.3 Usuários (atores de um banco de dados)

Um Banco de Dados pode apresentar diversos usuários cada qual com uma necessidade em especial, e com um envolvimento diferente com os dados. Os usuários podem ser classificados em categorias:

- **Administradores de bancos de dados (DBA):** em qualquer organização onde muitas pessoas compartilham muitos recursos, existe a ne-

cessidade de um administrador chefe para supervisionar e gerenciar estes recursos. Em um ambiente de base de dados, o recurso primário é a própria base de dados e os recursos secundários são o próprio SGBD e software relacionados. A administração desses recursos é de responsabilidade do DBA ("Database Administrator"). O DBA é responsável por autorizar acesso à base de dados e coordenar e monitorar seu uso, como também é responsável por problemas, tais como: quebra de segurança ou baixo desempenho. Em grandes organizações, existe uma equipe de DBAs.

- **Analistas de bancos de dados (projetistas):** possuem a responsabilidade de identificar os dados a serem armazenados no BD e pela escolha da estrutura apropriada utilizada para armazená-los. Eles devem se comunicar com os possíveis usuários do BD, obter a visão dos dados que cada um possui, integrando-as de forma a se obter uma representação adequada de todos os dados. Estas visões são então analisadas e, posteriormente, integradas para que, ao final, o projeto da base de dados possa ser capaz de dar suporte aos requisitos de todos os grupos de usuários.

- **Usuários finais:**existem profissionais que precisam ter acesso à base de dados para consultar, modificar e gerar relatórios. A base de dados existe para estes usuários. Podem ser usuários comuns do sistema, analistas de negócios ou usuários avançados que necessitam de uma maior interação com o BD.

- **Analistas de sistemas e programadores de aplicações:** os analistas determinam os requisitos dos usuários finais e desenvolvem especificações para transações que atendam estes requisitos, e os programadores implementam estas especificações, como: programas, testando, depurando, documentando e dando manutenção no mesmo. É importante que, tanto analistas quanto programadores, estejam a par dos recursos oferecidos pelo SGBD.

1.4 Níveis de abstração

A arquitetura ANSI/SPARC prevê múltiplas visões de dados, um esquema conceitual (lógico) e um esquema interno (físico). Um SBD se divide em geral em três níveis:

- **Nível externo:**possui as diversas descrições do BD de acordo com os grupos de usuários;

- **Nível conceitual:**descreve a estrutura de todo o BD para uma determinada comunidade de usuários, ocultando detalhes sobre a organização física dos dados e apresentando a descrição lógica dos dados e das ligações existentes entre eles;

- **Nível interno:**descreve a estrutura de armazenamento físico dos dados do BD, descreve o modelo físico dos dados que inclui detalhes sobre os caminhos de acesso aos dados internamente.

Figura 3: Níveis de Abstração

Sobre os níveis de abstração de um banco de dados podemos pensar assim: Nível externo é o que o usuário pensa e quer e deseja visualizar, nível conceitual é como o projetista irá implementar o banco de dados, e nível interno é como estes dados serão armazenados, formas de acesso físico por exemplo.

Observe que os três níveis apresentam apenas descrições dos dados. Como os três níveis apresentam descrições diferentes para os mesmos dados, torna-se necessário converter uma representação em outra, ou seja, definir mapeamentos de dados entre os níveis.

1.5 Sistemas Gerenciadores de Bancos de Dados - SGBD

Um Sistema de Gerenciamento de Bancos de Dados tem como principais propriedades:

- **Controle de redundância:** em um sistema tradicional de controle de arquivos cada usuário normalmente apresenta seus próprios arquivos armazenando o conjunto de dados que é de seu interesse, e nestes casos é comum ocorrer redundância de dados. Esta redundância consiste no armazenamento de uma mesma informação em locais diferentes, o que pode provocar sérios problemas. Alguns destes problemas consistem inicialmente no aumento de esforço computacional para realizar a atualização destes dados; aumento do espaço necessário para o armazenamento dos dados. O problema mais sério é que a representação dos dados desta forma pode tornar-se inconsistente, pois duas informações podem aparecer em locais distintos, mas apresentando valores diferentes. Em um sistema de

BD **as informações só se encontram armazenadas em um único local** ou estão existindo duplicação controlada dos dados.

- **Compartilhamento dos dados:**um SGBD deve incluir um software para o controle de concorrência ao acesso dos dados em um ambiente multiusuário, de forma que possibilite o compartilhamento dos dados, garantindo que se vários usuários tentem realizar operações de atualização sobre um mesmo conjunto de dados, o resultado destas operações possa ser correto e consistente.

- **Controle de acesso:** quando vários usuários compartilham os dados, é comum que alguns não apresentem autorização para acesso a todo o BD. Por exemplo, os dados do faturamento de uma empresa podem ser considerados confidenciais e, desse modo, somente pessoas autorizadas devem ter acesso. Além disso, pode ser permitido, a alguns usuários, apenas a recuperação dos dados. Já, para outros, são permitidas a recuperação e a modificação. Assim, o tipo de operação de acesso - recuperação ou modificação - pode também ser controlado. Tipicamente, usuários e grupos de usuários recebem uma conta protegida por senhas, que é usada para se obter acesso à base de dados, o que significa dizer que contas diferentes possuem restrições de acesso diferentes. Um SGBD deve fornecer um subsistema de autorização e segurança, que é usado pelo DBA para criar contas e especificar restrições nas contas. O SGBD deve então obrigar estas restrições automaticamente.

- **Possibilidade de múltiplas interfaces:**diversos usuários com níveis diferenciados de conhecimento técnico representam necessidades diversas no que se refere aos tipos de interfaces fornecidas pelo SGBD. Interfaces para consultas de dados, programação, e interfaces baseadas em menus ou em linguagem natural, são exemplos de alguns tipos que podem estar disponíveis. Um usuário com conhecimento técnico pode utilizar uma interface em que recupera dados mediante uma linguagem de consulta de dados, como SQL, já usuários com menor conhecimento técnico devem utilizar uma interface gráfica onde visualizam os dados e os selecionam para obter a consulta que necessitam ao invés de utilizarem uma linguagem para isto.

- **Representação de relacionamento complexo entre dados:**uma base de dados pode possuir uma variedade de dados que estão inter-relacionados de muitas formas. Um SGBD deve ter a capacidade de representar uma variedade de relacionamentos complexos entre dados, bem como recuperar e modificar dados relacionados de maneira fácil e eficiente.

- **Forçar restrições de integridade:** a maioria das aplicações de um banco de dados apresenta serviços que possibilitam garantir a integridade dos dados no BD. A restrição de integridade mais simples consiste na especificação do padrão de formato para os dados ou valores assumidos como um padrão. Como exemplo de restrição de integridade, podemos pensar quando queremos armazenar o nome e a idade de uma pessoa. O

nome deve ser uma cadeia de caracteres (string) menor que 50 caracteres alfabéticos, já a idade deve ser um dado numérico inteiro menor que 150. Estes são dois exemplos de restrições que podemos aplicar ao armazenar dados com intuito de garantir sua integridade.

- **Garantir backup e restauração de dados:**um SGBD deve prover recursos para realização de cópias de segurança e restauração caso ocorra falhas de hardware ou software. O subsistema de backup e restauração do SGBD é o responsável pela restauração. Por exemplo, se o sistema de computador falhar no meio da execução de um programa que esteja realizando uma alteração complexa na base de dados, o subsistema de restauração é responsável em assegurar que a base de dados seja restaurada no estado anterior ao início da execução do programa. Alternativamente, o subsistema de restauração poderia assegurar que o programa seja reexecutado a partir do ponto em que havia sido interrompido.

Vantagens de um SGBD

A escolha da tecnologia adequada de Banco de Dados e sua correta utilização trazem benefícios à maioria das organizações, tais como:

- **Potencial para garantir padrões:** a abordagem de base de dados permite que o Administrador do Banco (DBA) defina e force a padronização entre os usuários da base de dados em grandes organizações. Isso facilita a comunicação e a cooperação entre vários departamentos, projetos e usuários. Padrões podem ser definidos para: formatos de nomes, elementos de dados, telas, relatórios, terminologias, etc. O DBA pode obrigar a padronização em um ambiente de base de dados centralizado, muito mais facilmente que em um ambiente onde cada usuário ou grupo tem o controle de seus próprios arquivos e software.

- **Redução do tempo de desenvolvimento de aplicações:**um dos principais argumentos de venda para o uso da abordagem de um banco de dados é o tempo reduzido para o desenvolvimento de novas aplicações, tal como a recuperação de certos dados da base de dados para a impressão de novos relatórios. Projetar e implementar uma nova base de dados pode tomar mais tempo do que escrever uma simples aplicação de arquivos especializada. Porém, uma vez que a base de dados esteja em uso, geralmente o tempo para se criar novas aplicações, usando-se os recursos de um SGBD, é bastante reduzido. O tempo para se desenvolver uma nova aplicação em um SGBD é estimado em 1/4 a 1/6 do que o tempo de desenvolvimento, usando-se apenas o sistema de arquivos tradicional.

- **Independência de dados:**as aplicações de banco de dados não devem depender da forma como os dados estão representados e/ou armazenados.

- **Flexibilidade:**pode ser necessário alterar a estrutura de uma base de dados devido a mudanças nos requisitos. Por exemplo, um novo grupo

de usuários pode surgir com necessidade de informações adicionais, não disponíveis atualmente na base de dados. Um SGBD moderno permite que tais mudanças na estrutura da base de dados sejam realizadas sem afetar a maioria dos programas de aplicações existentes.

- **Disponibilidade para atualizar as informações:**um SGBD disponibiliza o banco de dados para todos os usuários. Imediatamente após um usuário modificar uma base de dados, todos os outros usuários "sentem" imediatamente esta modificação. Essa disponibilidade de informações atualizadas é essencial para muitas aplicações, tais como: sistemas de reservas de passagens aéreas ou bases de dados bancárias. Isso somente é possível devido ao subsistema de controle de concorrência e restauração do SGBD.

- **Economia de escala:**a abordagem de SGBDs permite a consolidação de dados e de aplicações, reduzindo-se, desse modo, o desperdício em atividades redundantes de processamento em diferentes projetos ou departamentos. Isto possibilita à organização como um todo investir em processadores mais poderosos e periféricos de armazenamento e de comunicação mais eficientes do que cada departamento adquirir seu próprio (menos potente) equipamento, o que reduz o custo total da operação e gerenciamento.

Apesar de todas as facilidades oferecidas por um banco de dados, um projeto de implantação pode gerar um alto custo inicial para a organização.

Linguagens

Uma vez que o projeto do BD tenha se completado e um determinado SGBD tenha sido escolhido para a sua implementação, o primeiro passo consiste em realizar uma especificação dos esquemas conceituais e internos, e os respectivos mapeamentos entre eles. Para estas etapas o SGBD oferece algumas linguagens apresentadas a seguir:

- **Linguagem de definição de dados (DDL - Data Definition Language):**utilizada pelos analistas e projetistas do BD para a definição dos esquemas do banco de dados. O SGBD também apresentará um interpretador para a DDL, o qual será responsável pelo processamento dos comandos da DDL, e realiza o armazenamento do esquema definido em estruturas internas do BD. Por exemplo, os comandos para criar, definir índice de uma tabela fazem parte da linguagem de definição de dados. Uma vez definido e preenchido o BD com os seus dados, estes normalmente sofrerão uma série de operações de acesso às informações nele armazenado.

- **Linguagem de manipulação de dados (DML):**o SGBD fornece esta linguagem para a especificação das operações de acesso ao banco. Os comandos da DML podem aparecer embutidos em outra linguagem (geralmente uma linguagem de programação de alto nível), e neste caso esta é denominada de Linguagem hospedeira, e DML é denominada de Sublinguagem de dados. De outra forma, se DML for utilizada isoladamente de

uma forma interativa, passa a ser denominada de Linguagem de consulta (ou "query language" como a SQL).

- **Linguagem de Controle de Dados (DCL):**no controle de acesso e transações dos dados utiliza-se esta linguagem, que inclusive possibilita estabelecer os diversos níveis de segurança de cada usuário.

Exemplos de SGBDs

Alguns exemplos de SGBDs encontrados no mercados são: MySQL, PostgreSQL, Microsoft SQL Server, MariaDB entre outros.

Link para Download do MySQL
Clique no link http://www.mysql.com/downloads/mysql/ para abrir o recurso.
Link para Download do PostgreSQL
https://www.postgresql.org/download/
Link para Download Microsoft SQL Server
https://www.microsoft.com/pt-br/sql-server/sql-server-downloads
Link para Download MariaDB
https://mariadb.org/download/

1.6 Vídeo complementar: Conceitos fundamentais sobre bancos de dados

Banco de Dados - Aula 1 - Parte 1 https://www.youtube.com/watch?v=qgnuH$_qSI9o$

1.7 Vídeo complementar 2: Conceitos fundamentais sobre bancos de dados

Banco de Dados - Aula 1 - Parte 2 https://www.youtube.com/watch?v=tEEAIs6aB2s

2 Modelos e modelagem

2.1 Modelos de dados

Os SGBDs evoluíram de sistemas de arquivos de armazenamento em disco, criando novas estruturas de dados com o objetivo de armazenar dados de forma mais eficiente e segura. Com o passar do tempo, os SGBDs passaram a utilizar diferentes formas de representação, ou modelos de dados, para descrever a estrutura das informações contidas em seus bancos de dados. A escolha do modelo de dados é a principal ferramenta no fornecimento de informações sobre a abstração realizada na parte de interesse específico no mundo real. Dentre os modelos existentes que normalmente são implementados pelos SGBDs podemos citar o modelo hierárquico, modelo em redes, modelo relacional (mais utilizado atualmente) além do modelo orientado a objetos.

2.1.1 Modelo hierárquico

O modelo hierárquico foi o primeiro modelo de dados. Nesse modelo de dados, os dados são estruturados em hierarquias ou árvores. Os nós das hierarquias contêm ocorrências de registros, onde cada registro é uma coleção de campos (atributos), cada um contendo apenas uma informação. O registro da hierarquia que precede a outros é o registro-pai, os outros são chamados de registros-filhos.

Uma ligação é uma associação entre dois registros. O relacionamento entre um registro-pai e vários registros-filhos possui cardinalidade 1:N (um para muitos, ou seja, um pai pode ter vários filhos). E há um tipo de registro chamado raiz, que não assume o papel de filho em nenhuma relação pai-filho. Os dados organizados conforme este modelo podem ser acessados como uma sequência hierárquica com uma navegação do topo para as folhas e da esquerda para a direita. Um registro pode estar associado a vários registros diferentes, desde que seja replicado. A replicação possui desvantagens: pode causar inconsistência de dados quando houver atualização e o desperdício de espaço é inevitável. Características:

- cada pai pode ter vários filhos;
- cada filho pode ter apenas um pai;
- duas entidades podem possuir apenas um relacionamento;
- qualquer recuperação de dados deve, obrigatoriamente, percorrer a estrutura da árvore. (pai à filho à neto...).

Como exemplos de banco de dados que utilizam o modelo hierárquico podemos citar: IMS e System 2.

Vamos tomar como exemplo uma agência bancária que, como pai de um relacionamento, pode ter vários clientes. O cliente, por sua vez, pode ter várias contas. Porém, se um cliente pertencer a mais de uma agência, seus dados terão que ser replicados, o que pode causar inconsistência e desperdício de espaço.

Figura 4: Exemplo de modelo hierárquico

2.1.2 Modelo em rede

O modelo em rede surgiu como uma extensão ao modelo hierárquico, eliminando o conceito de hierarquia e permitindo que um mesmo registro estivesse envolvido

em várias associações. No modelo em rede, os registros são organizados em grafos onde aparece um único tipo de associação que define uma relação 1:N (um para muitos) entre dois tipos de registros: proprietário e membro.

De modo contrário ao modelo hierárquico, em que qualquer acesso aos dados passa pela raiz, o modelo em rede possibilita acesso a qualquer nó da rede sem passar pela raiz, pois o modelo em rede permite a existência de entidades pais com muitos filhos e de entidades filhos com muitos pais.

Os modelos hierárquicos e em rede são orientados a registros, isto é, a qualquer acesso à base de dados como inserção, consulta, alteração ou remoção é feito em um registro de cada vez.

Tomando como exemplo a figura abaixo, um equipamento pode ter várias bombas e motores, e estes dois podem sofrer manutenção mecânica. Assim neste modelo perde-se a restrição hierárquica:

Figura 5: Exemplo de modelo em rede

São exemplos de SGBDs em rede: DBMS10, IDSII, DMSII e IMAGE

O modelo em rede apresenta como principal vantagem a possibilidade de uma modelagem mais próxima da realidade, porém não se firmou no mercado pelo surgimento do modelo relacional que veremos a seguir.

2.1.3 Modelo de dados orientado a objeto

Representam os dados como coleções que obedecem a propriedades. São modelos geralmente conceituais dispondo de poucas aplicações reais. Neste Modelo não seria interessante a existência de uma tabela de funcionários e dentro dela alguma referência para cada registro, de forma a podermos saber onde (em que departamento) o funcionário está alocado. Um conjunto de regras disponibilizaria em separado os funcionários da fábrica, que, no entanto, estariam agrupados aos demais, para o sistema de folha de pagamento.

2.2 Modelagem de dados

Utilizaremos para nossa modelagem o modelo Entidade Relacionamento, pois este é um modelo de dados conceitual de alto nível, cujos conceitos foram projetados para estar mais próximo possível da visão que o usuário tem dos dados, não se preocupando em representar como estes dados estarão realmente armazenados de maneira física. O modelo ER é utilizado principalmente durante o processo de projeto conceitual de banco de dados.

Quando pensamos no Modelo Entidade Relacionamento temos três conceitos básicos:

- Conjunto de Entidades: uma entidade pode ser vista como uma pessoa, "coisa" ou "objeto" no mundo real que é distinguível de todos os outros objetos, como por exemplo, um cliente de um banco. Já um conjunto é um grupo de entidades do mesmo tipo que compartilham os mesmos atributos, como por exemplo, um conjunto de clientes bancários, animais ou pessoas.

- Conjunto de Atributos: uma entidade é representada por um conjunto de atributos. Possíveis atributos do conjunto de entidades cliente, podem ser: nome-cliente, cpf, rua e cidade-cliente. Possíveis atributos do conjunto de entidade conta são: número-conta e saldo. Para cada atributo, existe um conjunto de valores permitidos chamado domínio daquele atributo. O domínio do atributo nome-cliente pode ser o conjunto de todas as cadeias de texto (strings) de certo tamanho.

  - Por exemplo, podemos limitar o tamanho do nome de uma pessoa a 50 caracteres, então este é seu domínio. Como também, o domínio do atributo número-conta pode ser o conjunto de todos os inteiros positivos. O atributo idade de uma entidade pessoa poderia ter como domínio os números inteiros entre 0 e 150.
  - Formalmente, um atributo é uma função que mapeia um conjunto de entidades em um domínio.

- Conjunto de Relacionamentos: um relacionamento é uma associação entre diversas entidades. Por exemplo, podemos definir um relacionamento que associa o cliente José Silva à conta 401. Isto especifica que José Silva é um cliente com conta bancária número 401. Um conjunto de relacionamentos é uma coleção de relacionamentos do mesmo tipo.

Nosso projeto conceitual de Banco de Dados será feito através da modelagem de dados usando o Modelo Entidade-Relacionamento (MER). Este modelo fornece as regras e conceitos para a criação de um Diagrama Entidade Relacionamento (DER), que deverá representar o banco de dados em questão, como por exemplo, um banco com vários clientes e contas, uma loja de produtos de beleza, de um consultório odontológico, uma indústria de peças, uma locadora de vídeo, uma escola, enfim, o negócio a que este banco de dados deve servir. Desta maneira, usaremos dois modelos teóricos para a construção de um banco de dados:

- o modelo entidade relacionamento para criação do projeto conceitual do banco de dados;
- o modelo relacional para a implementação em um ambiente computacional pré-existente. Este ambiente computacional pré-existente será o nosso Sistema Gerenciador de Banco de Dados Relacional - MySQL.

Com intuito de realizar a transferência entre modelos serão aplicadas as regras de mapeamento do Modelo Entidade Relacionamento ME-R para o Modelo Relacional MRel. Ou seja, para "converter" o diagrama conceitual de nosso banco de dados em um conjunto de tabelas relacionadas que possa ser implementado em um SGBD Relacional, usaremos um conjunto de passos. Este processo é chamado mapeamento do MER para o MRel.

Para realizarmos o mapeamento dos dados entre os modelos, primeiramente utilizaremos os requisitos coletados para construção e realizaremos a modelagem utilizando o Modelo Entidade Relacionamento, em seguida "converteremos" os diagramas gerados em tabelas sobre o modelo relacional para implementarmos em um SGBD, onde teremos os seguintes passos:

1. Requisitos para a base de dados;
2. Modelo Entidade Relacionamento;
3. Modelo Relacional;
4. SGBD Relacional.

Agora veremos como construir um diagrama Entidade Relacionamento DER para o projeto de um banco de dados. Este será o primeiro passo para a construção de nosso banco de dados. O Modelo Entidade Relacionamento MER é composto por Entidades e Relacionamentos sem nos esquecermos dos atributos.

Representações

A estrutura lógica geral de um banco de dados pode ser expressa graficamente por um diagrama ER, que consiste nos seguintes componentes:

1. **Retângulos que representam conjuntos de entidades;**
2. **Elipses** que representam atributos;
3. **Losângos** que representam relacionamentos entre conjuntos de entidades;
4. **Linhas** que ligam atributos a conjuntos de entidades e conjuntos de entidades a relacionamentos.

Alguns autores chamam as linhas de arestas, em analogia às teorias de grafos e redes.

Entidades e atributos

Como descrito acima, a representação para um conjunto de entidades é um retângulo e, para cada atributo, uma elipse, como por exemplo, um conjunto de entidades Aluno pode possuir os atributos nome, prontuário e data de nascimento:

Figura 6: Entidade com seus atributos)

Este conjunto de entidades ALUNO possui os conjuntos de atributos Nome, Prontuario e Data_Nasc. Uma entidade desse conjunto poderia ser ('Iago', 10123, '2000-12-16').

Chave de um conjunto de entidades

É importante poder especificar como entidades e relacionamentos são identificados. Conceitualmente, entidades e relacionamentos individuais são distintos, mas em uma perspectiva de banco de dados a diferença entre eles precisa ser expressa em termos de seus atributos. Uma restrição importante sobre entidades é a aplicação de uma chave única. Um conjunto de entidades sempre possui um atributo cujo valor é diferente e válido para cada entidade. Tal atributo é chamado atributo-chave, e seu valor é usado para identificar cada entidade de modo unívoco, único, como por exemplo, o atributo CPF de uma pessoa é sempre único, ou o prontuário de um aluno é sempre único. A chave de um atributo será sempre sublinhada. Assim sendo, todo conjunto de entidades deve ter um conjunto de chaves cujo valor identifique com unicidade a entidade, pois a chave é o principal meio de acesso a uma entidade.

É possível ocorrer situações onde é preciso mais de um atributo para identificar cada entidade do conjunto. Por exemplo, o RG (registro geral) dos brasileiros não identifica um único cidadão. É possível, que um mesmo número de RG possa ocorrer em duas unidades diferentes da federação, ou seja, um mesmo número de RG poderia ser emitido em São Paulo e Minas Gerais. Assim, para ser identificador, o RG precisa ser composto com o valor do Órgão Emissor, no caso de São Paulo, SSP_SP. Estes dois atributos (RG e Órgão Emissor) se, juntos, identificam cada brasileiro. Nestes casos, quando é necessário mais de um atributo, a chave do conjunto de entidades será composta.

2.2.1 Relacionamentos

Nenhuma informação armazenada no Banco de Dados existe isoladamente. Todos os elementos pertencentes ao mundo real (restrito) modelado de alguma forma estão associados a outros elementos. Normalmente estas associações representam ações físicas ou alguma forma de dependência entre os elementos envolvidos. Um relacionamento é uma associação entre diversas entidades.

Considere um conjunto de relacionamentos TRABALHA entre os conjuntos de entidades: EMPREGADO e DEPARTAMENTO. Este relacionamento associa cada empregado com o departamento em que ele trabalha. Cada instância de relacionamento em TRABALHA associa uma entidade "empregado" a uma entidade "departamento". Cada instância de relacionamento conecta uma entidade EMPREGADO a uma entidade DEPARTAMENTO.

O conjunto de relacionamentos é, portanto, representado por um losango. Enquanto que para os conjuntos de entidades os atributos são obrigatórios, para os conjuntos de relacionamentos, eles são optativos. O conjunto de entidade só faz sentido quando especificamos seus atributos.

Já um conjunto de relacionamentos (CR), nem sempre precisa possuir atributos. Sua existência justifica-se apenas pela função de relacionar uma ou mais entidades. Em geral, os atributos dos conjuntos de relacionamentos, quando existem, especificam dados sobre tempo (data, horário), quantidades, valores, enfim, atributos relativos a transações, ações, ocorrências, que caracterizam os relacionamentos.

Grau de relacionamento

O grau de um conjunto de relacionamentos indica o número de conjuntos de entidades participantes. Um tipo de relacionamento de grau dois é chamado binário, de grau três de ternário, de grau quatro quaternário, acima disso, n-ário. A quantidade de Entidades envolvidas em um Relacionamento pode ser determinada por sua semântica. Desta forma, podem-se categorizar os graus de relacionamento em:

- **Unário:** é o grau de Relacionamento que envolve um único Tipo de Entidade.

Figura 7: Exemplo de relacionamento unário

- **Binário:** é o grau de Relacionamento que envolve dois Tipos de Entidades.

Figura 8: Exemplo de relacionamento binário

- **Ternário:** é o grau de Relacionamento que envolve três Tipos de Entidades.

Figura 9: Exemplo de Relacionamento ternário

- **Quaternário:** é o grau de Relacionamento que envolve quatro Tipos de Entidades.

Figura 10: Exemplo de relacionamento quaternário

A quantidade de Entidades envolvidas em cada Relacionamento é determinada pela Cardinalidade do grau de Relacionamento, ou seja, podemos estabelecer a quantidade mínima e máxima de Entidades envolvidas com cada Entidade relacionada.

- A **Cardinalidade Mínima** determina a quantidade mínima de Entidades relacionadas por um número representativo, ou seja, 0 (zero) 1, 2, ..., N (muitos);
- A **Cardinalidade Máxima** determina a quantidade máxima de Entidades relacionadas por um número representativo, ou seja, 1, 2, ..., N (muitos).

Restrições de relacionamento

Os relacionamentos entre entidades possuem certas restrições que limitam as combinações possíveis das entidades que dele participam. Uma destas restrições é a razão de **cardinalidade**. A quantidade de Entidades envolvidas em cada Relacionamento é determinada pela Cardinalidade do Tipo de Relacionamento, ou seja, pode-se estabelecer a quantidade mínima e máxima de Entidades envolvidas com cada Entidade relacionada.

Até agora vimos, de modo geral, como construir um diagrama de entidade relacionamento (DER) para projetar um banco de dados. Aprendemos os quatro principais construtores deste modelo: os conjuntos de entidades (CE), os conjuntos de relacionamentos (CR), os atributos de entidades e relacionamento. Agora, veremos um dos mais importantes conceitos do MER: a restrição de razão de cardinalidade do relacionamento. O entendimento deste conceito será imprescindível para o processo de mapeamento do diagrama entidade relacionamento DER para o Modelo Relacional (MRel), ou seja, para transformar o projeto conceitual do banco de dados em um conjunto de tabelas.

2.2.2 Link: Software DIAURL

Click http://dia-installer.de/ link to open resource.

3 Modelo de dados relacional

3.1 Introdução

Os SGBDs que apresentaram maior sucesso e confiabilidade no mercado são os que adotaram o Modelo Relacional como base para sua construção. O Modelo Relacional, de modo geral, permite que os dados sejam "vistos" como tabelas. Ora, esta é uma maneira muito natural de armazenar e recuperar dados. Geralmente, quando fazemos uma relação de dados, tendemos a organizá-los em linhas e colunas, no formato de tabelas. Esta é uma importante razão para o sucesso das planilhas eletrônicas no mundo dos negócios.

O modelo relacional surgiu devido às necessidades de **aumentar a independência de dados** nos sistemas gerenciadores de banco de dados, proporcionando um conjunto de funcionalidades apoiadas em álgebra relacional para armazenamento e recuperação de dados além de permitir processamento dedicado. O modelo relacional revelou-se ser o mais **flexível** e adequado ao solucionar os vários problemas que se colocam no nível da concepção e implementação da base de dados. A estrutura fundamental do modelo relacional é a relação (tabela). Uma relação é constituída por um ou mais atributos (campos) que traduzem o tipo de dados a armazenar. Cada instância do esquema (linha) é chamada de tupla (registro). Este modelo não possui caminhos pré-definidos para se fazer acesso aos dados como nos modelos anteriores. O modelo relacional implementa estruturas de dados organizadas em relações, porém, para trabalhar com essas tabelas, algumas restrições precisaram ser impostas para evitar redundância, perda de dados e incapacidade de representar parte da informação. Essas restrições são: integridade referencial, chaves e integridade de junções de relações.

Atualmente, mesmo que informalmente, o Modelo Relacional é visto no mercado quase como um sinônimo de Sistemas de Banco de Dados. Normalmente, quando nos referimos a banco de dados, os usuários já imaginam um conjunto de tabelas relacionadas.

3.2 Conceitos

O Modelo Relacional será a base para implementarmos nosso banco de dados. Ele representa os dados da base de dados como uma coleção de relações. Podemos pensar que, cada relação pode ser entendida como uma tabela ou um simples arquivo de registros (Citação). Por exemplo, a base de dados de arquivos que são mostradas nas tabelas abaixo, é similar a representação do modelo relacional. No entanto, possui algumas importantes diferenças entre relações e arquivos como veremos a seguir.

Nome	Número	Turma	Curso_Hab
Smith	17	1	CC
Brown	8	2	CC

NomedoCurso	Numerodocurso	Créditos	Departamento
Introdução à Ciência da Computação	CC1310	4	CC
Estrutura de Dados	CC3320	4	CC
Matemática Discreta	MAT2410	3	MATH
Banco de Dados	CC3380	3	CC

NumerodoCurso	NumerodoPre_requisito
CC3380	CC3320
CC3380	MAT2410
CC3320	CC1310

Ao pensarmos em uma relação como uma tabela de valores, cada linha representa uma coleção de valores que estão relacionados. Esses valores podem ser interpretados como um fato que descreve uma entidade ou uma instância. Utilizamos o nome da tabela e os nomes das colunas para nos ajudar a interpretar o significado dos valores em cada linha da tabela, que são na verdade, dados a respeito dos dados, chamados **metadados**.

Observe que a primeira tabela é traz dados referentes a alunos porque cada linha representa fatos sobre uma entidade aluno em particular. Os nomes das colunas - Nome, Número, Turma, Departamento - especificam como interpretar os valores em cada linha baseando-se nas colunas que cada valor se encontra.

Na linguagem do modelo relacional, cada linha é chamada de **tupla**, a coluna ou cabeçalho é chamado de **atributo** e a tabela de **relação**. Desta maneira, o conjunto de nomes das tabelas e suas colunas são chamados de esquema da relação. Assim, o esquema da relação Aluno é:

- ALUNO = nome, número, turma, departamento

Temos que conhecer também conceito de **grau de uma relação**, este é o número de atributos da relação. No exemplo acima, o grau de relação do esquema Aluno é quatro, pois possui quatro colunas.

Uma coluna de dados possui um tipo de dado que descreve os valores que podem aparecer nela, por exemplo, na coluna **Númer**o de um aluno esperamos um valor numérico como 17, 18 e não caracteres, este tipo de dado que especifica os possíveis valores de uma coluna é chamada de **domínio**. No esquema de relação Aluno podemos especificar alguns domínios para atributos da relação Aluno, vejamos alguns exemplos:

- **Nome**: Conjunto de cadeia de caracteres que representa nomes de pessoas;
- **Número**: Conjunto de dados numéricos com limite de cinco dígitos;

- **Turma**: Conjuntos de códigos das turmas da faculdade;
- **Departamento**: Conjunto de códigos dos departamentos acadêmicos, como CC, EP, etc.

Assim, de acordo com nosso exemplo de domínio para a tabela Aluno, o que esperamos obter em uma linha (tupla), é o conjunto de valores dos atributos para um determinado estudante. Por exemplo, existe um aluno de nome Smith, seu número é 17, sua turma é 1 e seu departamento CC.

A tabela abaixo mostra um exemplo da relação Aluno. Cada linha (tupla) representa uma entidade aluno em particular, cada coluna (atributo) corresponde a um cabeçalho de coluna, os valores apresentados como nulos (null) representam atributos em que os valores não existem para alguma tupla individual de Aluno:

Nome	**INSS**	**FoneResidencia**	**Endereco**	**FoneEscritorio**	**Idade**
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	*null*	19
Katherine Ashly	381-62-1245	375-4409 1	125 Kirby Road	*null*	18
Dick Davidson	422-11-2320	*null*	3452 Elgin Road	749-1253	25

3.3 Atributos-chaves

Uma relação é definida como um conjunto de tuplas. Por padrão, todos os elementos de um conjunto devem ser distintos. Assim, todas as tuplas de uma relação também são distintas. Isto significa que nenhuma tupla pode ter a mesma combinação de valores para todos os seus atributos. Desta maneira, temos que ter um valor que chamamos de **atributo-chave** que é utilizado para identificar de modo unívoco uma tupla em uma relação.

Geralmente, um esquema de relação pode ter mais que uma chave. Nos casos em que isto ocorra, cada chave é chamada **chave-candidata**. Por exemplo, o esquema da relação Aluno poderia ter um atributo adicional Código, para indicar o código interno de alunos na escola. Assim, o esquema teria duas chaves candidatas: Número e Código.

Chave-Primária

Após identificarmos as chaves candidatas devemos definir uma delas como a **chave-primária** da relação. A indicação no modelo de qual chave-candidata é a chave-primária é realizada se sublinhado ou negritado os atributos que formam a chave-candidata escolhida, como podemos ver abaixo:

Figura 11: Representação da tabela "aluno"

O atributo Nome da relação Aluno não deve ser indicado como chave, uma vez que nada garante a que não haja ocorrência de nomes duplicados (homônimos).

Em certas relações pode ser necessário mais de um atributo para identificar cada tupla da relação de forma unívoca. Por exemplo, vejamos a relação Pessoa apresentada abaixo:

Figura 12: Representação da relação pessoa

Um determinado atributo pode não garantir a unicidade de uma tupla, desta maneira a identificação de cada pessoa deve ser feita pelo valor do atributo RG e do atributo OrgaoEmissor. A utilização de mais de um atributo para a composição da chave primária chamamos de chave composta. Alternativamente para não utilizarmos uma chave composta na relação Pessoa, poderíamos incluir um campo de identificação única como um código, por exemplo.

Chave estrangeira

O conceito de chave estrangeira é de grande importância na construção de banco de dados relacional. Vamos começar com exemplo:

Figura 13: Representação de um conceito de chave estrangeira

No esquema acima temos três relações. A relação FUNCIONÁRIO possui os dados de um funcionário de uma empresa, como o Nome, CPF, Setor onde trabalha e Data de Nascimento. A relação SETOR possui o Nome e Localização de um setor que é identificada por um código. A relação DEPENDENTE possui o CPF deste como chave primária, o CPF do funcionário o qual ele é dependente, o Nome e a Data de Nascimento do dependente.

Podemos verificar que alguns atributos estão presentes em mais de uma tabela. Através das relações apresentadas abaixo podemos verificar que o atributo Setor da tabela FUNCIONARIO representa o código do setor onde o funcionário está lotado, sendo o mesmo atributo Cod_Setor da tabela SETOR. O mesmo caso ocorre entre os atributos CPF e CPF_Funcionario das tabelas FUNCIONARIO e DEPENDENTE respectivamente.

Figura 14: Diagrama ER das tabelas Funcionário, Setor e Dependente

Relação entre tabelas

Como podemos verificar, apenas a chave primária de uma tabela deve ser repetida em outra tabela. É o que acontece no esquema relacional apresentado. A chave primária de FUNCIONARIO está representada na tabela DEPENDENTE e a chave primária da tabela SETOR está representada na tabela FUNCIONARIO, como **chave estrangeira**. Desta maneira, podemos saber, por exemplo, qual é o SETOR de um funcionário (através do atributo Setor) ou quais dependentes possui um FUNCIONARIO através do atributo CPF_Funcionario.

Desta forma, a **chave estrangeira** possibilita a implementação do conceito de relacionamento entre entidades. Com isto, podemos no Modelo Relacional, realizar a modelagem representada no Modelo Entidade Relacionamento que mostra um conjunto de entidades relacionado a outro conjunto de entidades,

com determinada razão de cardinalidade.

3.4 Integridade referencial

A **restrição de integridade referencial** é responsável por manter a consistência entre tuplas (registros) de duas relações. Esta restrição de integridade referencial estabelece que uma tupla de uma relação que se refere à outra relação deve se referir a uma tupla existente naquela relação. Vamos exemplificar para ficar mais claro, vejamos duas tabelas, a primeira sobre funcionários e a segunda, setor:

NOME	CPF	SETOR	DATA_NASC
Jose	33875419294	1	10/05/1980
Carlos	26218188892	2	17/05/1970

Cod_Setor	Nome	Localizacao
1	Informatica	Andar 4
2	RH	Térreo

Nas tabelas podemos observar que o atributo Setor de FUNCIONARIO indica o número do SETOR que cada funcionário trabalha. Assim, todos os valores do atributo Setor (chave estrangeira) nas tuplas da relação FUCIONARIO devem pertencer ao conjunto de valores do atributo Cod_Setor (chave primária) da relação SETOR. Desta maneira podemos definir que o conceito de Integridade Referencial decorre da implementação de chaves estrangeiras em um esquema relacional. Assim temos duas regras para dizer que um conjunto de atributos será chave estrangeira de uma relação:

- Os atributos da **chave estrangeira** têm o mesmo **domínio** dos atributos da **chave-primária** a qual se relaciona. Podemos dizer então que os atributos chave estrangeira fazem referência à chave primária;
- O valor da **chave estrangeira** será algum valor que ocorre na **chave primária** a que ela faz referência ou terá o valor null.

As restrições de integridade devem ser especificadas no esquema da base de dados relacional se o projetista quiser manter essas restrições válidas para toda a base de dados. Desta maneira, em um sistema relacional, a linguagem de definição de dados (DDL) deve fornecer métodos para especificar os vários tipos de restrições tal que o SGDB possa garanti-las automaticamente.

3.5 Mapeamento

O Modelo Relacional (MRel) implementa as tabelas relacionadas, com muitos recursos para segurança dos dados, controle de acesso, consultas e manutenção

dos dados. Porém, este modelo não é o modelo mais adequado para se fazer projetos de banco de dados. Então conhecemos o Modelo Entidade Relacionamento MER, que por meio de seus recursos visuais, se apresenta mais claro, simples e intuitivo.

Desta maneira, em projetos de banco de dados, normalmente a modelagem dos dados é realizada através de um modelo de dados de alto-nível. O modelo de dados de alto-nível geralmente adotado é o Modelo Entidade- -Relacionamento (MER) e o esquema das visões e de toda a base de dados é especificado em diagramas entidade-relacionamento (DER).

O Modelo Entidade Relacionamento possui um maior número de abstrações. Existem ainda definições para: Conjunto de Entidades Fracas, Atributos Compostos, Atributos Multivalorados, por exemplo. Vimos os principais conceitos do modelo, que permitem a construção da maior parte dos bancos de dados para as aplicações comerciais mais comuns.

Agora, veremos como traduzir um DER para um esquema MRel. Para isto utilizaremos um conjunto de passos para que possamos implementar nosso banco de dados em um SGBD Relacional.

- **Passo 1: Mapeamento dos tipos de entidades:**Consiste na criação de uma relação que inclua todos os atributos de uma entidade. Assim, as entidades do MER são transformadas em tabelas no MRel. O atributo chave da entidade passa a ser a chave primária da relação (tabela).

- **Passo 2: Mapeamento dos Conjuntos de Relacionamentos binários de razão de cardinalidade 1:1:** Para este caso temos 3 possíveis opções de mapeamento, porém, as opções mais utilizadas e que devem ser seguidas são a primeira e a terceira.:

  1. criar chave estrangeira em uma das relações;
  2. gerar uma única relação para as entidades e o relacionamento; e
  3. gerar uma relação exclusiva para o relacionamento, porém, esta opção caracteriza como veremos mais adiante o mapeamento N:M.

- **Passo 3: Mapeamento dos Conjuntos de Relacionamentos binários de razão de cardinalidade 1:N:**Em primeiro , devemos gerar uma tabela para cada um dos conjuntos de entidades conforme descrito em nosso primeiro passo. Para o mapeamento 1:N a relação que mapeia o Conjunto de entidades do lado N recebe a chave primária do outro Conjunto de Entidades (lado 1) como chave estrangeira e os atributos do relacionamento.

- **Passo 4: Mapeamento dos Conjuntos de Relacionamentos binários de razão de cardinalidade N:M:**Devemos criar uma nova tabela para representar o relacionamento. Nesta nova tabela devemos incluir como chave- -estrangeira as chaves-primárias das tabelas que representam os tipos de entidade participantes; sua combinação irá formar a chave-primária desta nova tabela.

- **Passo 5: Mapeamento de Relacionamentos N-ários:** Para os relacionamentos de grau maior que dois devemos criar uma nova relação (tabela) para representar o relacionamento. Temos que incluir nesta tabela como chave- -estrangeiras as chaves-primárias das relações que representam os tipos de entidades participantes. Temos que incluir também qualquer atributo simples do tipo de relacionamento n-ário. A chave-primária desta relação é normalmente uma combinação de todas as chaves-estrangeiras fazendo referência às relações que representam os tipos de entidades do relacionamento.

4 Normalização de dados

4.1 Introdução

Agora aprenderemos a montar o Modelo Entidade Relacionamento (MER), considerando outro ponto de vista. Anteriormente, era apresentado o contexto em que tinha-se a ideia de modelar o banco e eram definidas as entidades, relacionamentos e atributos de forma bastante subjetiva. A normalização de dados trata da construção do modelo de dados, mas considera fatos reais, como documentos fiscais, formulários, planilhas, fichas, entre outras coisas. A ideia básica é que a partir destes documentos, aplicando regras previstas no processo de normalização, seja gerado o Modelo de Dados.

Para atingir os objetivos, podemos dizer que um banco de dados está relacionado a alguns conceitos fundamentais:

- Banco de Dados = Coleção de Arquivos;
- Arquivos = Coleção de Registros;
- Registro = Coleção de Campos (Tuplas);
- Campo = Coleção de Caracteres;
- Caractere = Alfa-numéricos ou símbolos.

De acordo com Machado, existem nove técnicas que podem ser utilizadas no desenvolvimento de um banco de dados nas fases iniciais, as quais podemos citar:

1. Contextualização;
2. Objetivação;
3. Intitulação;
4. Especificação de Requisitos;
5. Normalização;

6. Modelagem;
7. Trigramação;
8. Dicionarização;
9. Auditoria das fases de Análise e de Projeto Lógico do Sistema de Banco de Dados.

Agora, estaremos focados no que diz respeito à Normalização como ferramenta para o desenvolvimento de um banco de dados.

4.2 Formas normais

A Teoria da Normalização é expressa tradicionalmente através de um conjunto de Formas Normais, que otimizam a estrutura e o conteúdo das relações entre as entidades.

O conceito de normalização foi introduzido por E. F. Codd, em 1970, utilizando a primeira forma normal – 1FN. Esta técnica, pode-se dizer, é um processo matemático formal que fundamenta-se na teoria dos conjuntos.

Nesse processo podem ocorrer as seguintes Anomalias de Atualização:

- Inclusão: ao se incluir um novo cliente, por exemplo, ele tem que estar relacionado com uma venda;
- Alteração: caso o fabricante de um produto altere o preço de determinada classe de produto, será preciso percorrer toda a Entidade para se realizarem múltiplas alterações;
- Exclusão: ao se excluir um cliente, por exemplo, os dados referentes às suas compras poderão ser perdidos.

Através do processo de normalização podemos substituir, de forma gradual, um conjunto de Entidades e Relacionamentos para um modelo mais "adequado", em relação às Anomalias de Atualização (Inclusão, Alteração e Exclusão), as quais podem causar certos problemas:

- Grupos repetitivos (atributos multivalorados) de dados;
- Dependências parciais em relação a uma chave concatenada;
- Redundâncias de dados desnecessários;
- Perdas acidentais de informação;
- Dificuldade na representação de fatos da realidade observada;
- Dependências transitivas entre atributos.

Estes problemas podem ser minimizados através da aplicação da normalização, tornando o modelo de dados bastante estável e sujeito a poucas manutenções.

4.3 Dependência funcional

Para descrever as formas normais, faz-se necessária a introdução deste conceito importante na aplicação da normalização, no qual a maior parte da teoria de normalização foi baseada. Dada uma entidade qualquer, dizemos que o atributo ou conjunto de atributos de A é dependente funcionalmente de outro atributo de B; neste caso, há dependência funcional.

Em outras palavras, podemos dizer que, em uma entidade TB_ITENSNOTA, o atributo ITN_CODPRODUTO depende funcionalmente do atributo PRO_CODIGO, que encontra-se na entidade TB_PRODUTO.

A análise das relações existentes entre os atributos de uma entidade deve ser feita a partir de um conhecimento prévio da sistemática aplicada nas relações entre as entidades, para que esta possa ser modelada, como no exemplo anterior.

Para iniciar a normalização, faz-se necessária a descrição de todos os atributos do documento que se deseja normalizar (visando a criação do modelo de dados), para que posteriormente, possam ser aplicadas técnicas (formas normais) que irão dividir os atributos em entidades, as quais darão suporte à construção do modelo de dados.

4.4 Normalização de dados (nota fiscal)

Supondo a necessidade de uma empresa em modelar um banco de dados para emissão de uma nota fiscal simples, como apresentado no quadro abaixo. Verifique que há dados que são específicos da empresa, representados aqui como "Dados da Empresa"e que não há necessidade de manter-se no banco de dados, pois são impressos diretamente na nota fiscal encaminhada pela gráfica. O que nos interessa aqui é justamente os dados que necessitam ser preenchidos, como Data, Número da Nota Fiscal, Cliente, RG, CPF, Cidade, UF, Endereço, Número e Bairro, além dos dados dos produtos vendidos, inerentes ao Código do produto, Descrição, Unidade de Medida, Quantidade, Valor Unitário e Valor Total.

Dados da Empresa			Data: 01/01/0101		NF Nº 123
CLIENTE: João da Silva				RG: 000000000-00	
CPF: 000000000-00		CIDADE: Medianeira			UF: PR
ENDEREÇO: R. Bahia		Nº 1234	BAIRRO: Centro		
Código	Descrição	Und. de Medida	Quantidade	Vlr Unitário	Vlr Total
1	Arroz	kg	2	1,99	3,98
2	Sabonete	Unidade	1	0,79	0,79
3	Leite	Litro	3	2,12	6,36
				Valor TOTAL: 11,13	

Figura 15: Modelo de nota fiscal

Inicialmente, realiza-se a definição de todos os atributos que o documento

possui, registrando que eles são tidos como atributos da entidade principal (documento), conforme segue.

Neste caso, a entidade principal foi definida como Nota e a ela foram relacionados todos os atributos oriundos a emissão de uma nota fiscal. Foi definida ainda a chave primária, única para cada nota fiscal a ser emitida: **Nota** (**número da nota**, data, cliente, RG, CPF, cidade, UF (Unidade da Federação), endereço, nº, bairro, código_produto_1, código_produto_2, código_produto_3, ..., descrição_produto_1, descrição_produto_2, descrição_produto_3,..., unidade_produto_1, unidade_produto_2, unidade_produto_3, quantidade_produto_1, quantidade_produto_2, quantidade_produto_3, ..., valor_unitário_produto_1, valor_unitário_produto_2, valor_unitário_produto_3 ,..., valor_total_produto_1, valor_total_produto_2, valor_total_produto_3,..., valor_total_da_nota);

- O campo em negrito indica a chave primária.

Aplicação da 1ª Forma Normal – 1FN

Existem situações em que nos deparamos com algumas informações que se repetem (atributo multivalorado) dentro de uma única linha, ligada a uma chave primária.

A 1FN diz que cada ocorrência da chave primária deve corresponder a somente uma informação de cada atributo, ou seja, a entidade não deve conter atributos repetidos ou multivalorados, ou, ainda, os atributos não-chave deverão ser atômicos (únicos).

Ao observarmos que certo grupo de atributos não-chave não são atômicos, ao longo do processo de entrada de dados, verificamos que este atributo deverá ser decomposto em uma nova entidade.

Nas novas entidades criadas, a chave primária será a concatenação da chave primária da entidade original, o que mantém, desta forma, o relacionamento entre elas, como também o conjunto de atributos que se repetem.

Por exemplo: a 1FN diz que todos os atributos devem conter um valor atômico (único), ou seja, os dados inclusos dentro do campo não devem se repetir.

Na prática: o campo cliente vai ser preenchido quantas vezes na nota acima? Apenas uma vez, certo? Então corresponde a um valor atômico. Já os campos que se referem aos itens da nota serão acrescidos uma ou muitas vezes. Desta forma, tem-se a necessidade de separação da entidade Nota, criando-se a nova entidade chamada Itens_Nota, ficando:

Nota (número da nota, data, cliente, RG, CPF, endereço, nº, bairro, cidade, UF, valor_total).

Os campos que se repetem são destinados a uma nova tabela, no nosso exemplo chamado de itens da nota. A chave principal também é inclusa na nova tabela para manter o relacionamento entre elas.

Itens_Nota (número da nota, código, descrição, unidade_de_medida, valor_unitário, quantidade, valor_total).

- Note: as tabelas que saem na 1FN são tabelas associativas.

Aplicação da 2ª Forma Normal – 2FN

A 2FN diz que todos os atributos que não forem chave têm que ser dependentes diretos da chave, ou seja, os atributos que estão contidos na nota têm que depender diretamente da chave **Número da nota**.

Inicialmente, observa-se a entidade que possui chave primária concatenada. Para aquelas que satisfazem esta condição, analisar se existe algum atributo ou conjunto de atributos com dependência direta à chave primária.

Exemplo: o atributo cliente é dependente direto da nota, pois é o cliente desta nota. Já atributo RG, CPF são dependentes do cliente, não tendo nenhuma relação direta com a nota, ou seja, independentemente da nota, o CPF do cliente será o mesmo se soubermos o nome do cliente. Sabe-se também que o CPF do cliente nunca muda. Desta forma, tem-se:

- **Nota (número da nota**, data, código_cliente, endereço, nº, bairro, código_cidade, valor_total);
-
  - Os atributos adicionados acima "código_cliente e código_cidade" são para manter o relacionamento entre as entidades.
  - Verifique que o campo endereço deve permanecer na entidade Nota, tendo em vista que, se, por acaso, pretender-se buscar uma nota emitida há cinco anos cujo cliente alterou seu endereço para outra rua, o endereço que deverá aparecer na nota emitida há cinco anos deve ser o correspondente ao endereço antigo do cliente, e não o novo endereço.
- **Cliente (código**, nome, RG, CPF);
  - Os campos dependentes do cliente são transferidos para a tabela cliente.
- **Cidade (código**, descrição, UF);
  - O atributo UF que depende apenas da cidade é atribuído para a tabela cidade.
- **Itens nota (número_da_nota, código_produto**, quantidade, valor_unitário, valor_total);
  - O atributo "código_produto" é mantido na tabela associativa para manter o relacionamento entre Nota, Itens da Nota e Produtos.
- **Produto (código**, descrição, unidade_de_medida, valor_unitário);
  - Os atributos dependentes de produto são relacionados tabela produto. Repare que o atributo valor unitário foi duplicado, isso pela necessidade do valor unitário de um produto sofrer mudanças, em notas diferentes o valor unitário pode estar diferente também.

Aplicação da 3ª Forma Normal – 3FN

A 3FN diz que o atributo precisa estar na segunda forma normal, e todos os atributos que não são chave, não podem depender de outros atributos que também não são chave. Deve-se verificar ainda se o valor de um atributo pode se repetir em muitas tuplas, sendo que neste caso, é necessário inclusão de nova entidade.

Uma Entidade está na 3FN se nenhum de seus atributos possui dependência em relação a outros atributos da Entidade. Ao verificar a existência de dependência entre as Entidades e seus atributos, devemos criar outra entidade que contenha os atributos dependentes.

- **Nota (número da nota**, data, código_cliente, código_endereço, código_bairro, nº, código_cidade);
  - Note que o atributo valor total da nota foi retirado na 3FN. Todos os atributos que são gerados a partir do resultado de dois ou mais atributos são retirados, visando manter a consistência dos dados.
- **Endereço (código**, descrição);
  - Aplicando a 3FN, o endereço é retirado da tabela nota, pois o endereço não é um atributo chave e refere-se ao endereço do cliente nesta nota. Então, o endereço é incluso na tabela de clientes e relaciona-se tanto com a entidade Cliente como a entidade Nota.
- **Bairro (código**, descrição);
  - Aplicando a 3FN, o bairro é retirado da tabela nota, pois o bairro não é um atributo chave e refere-se ao bairro do cliente nesta nota. Então, o bairro é incluso também na tabela de clientes e relaciona-se tanto com a entidade Cliente como a entidade Nota.
- **Cliente (código**, nome, RG, CPF, nº, **código_endereço, código_cidade)**;
  - Os campos chave código_endereço e código_cidade vêm para a tabela de clientes com o fim de fazer o relacionamento entre as tabelas.
- **Cidade (código**, descrição, UF);
- **Itens nota (número da nota, código_produto**, quantidade, valor_unitário);
  - O atributo valor total sai, pois ele é resultado de outros dois atributos: a quantidade vezes o valor unitário.
- **Produto (código**, descrição, unidade_de_medida, valor_unitário).

4.5 Normalização de dados (formulário remanejamento)

Dado o formulário de solicitação de remanejamento, será aplicada a normalização de dados visando obter um modelo de dados conciso.

Definição dos atributos contidos no formulário

Requerimento (código, nome, sexo, data_nascimento, categoria_funcional, lotação_atual, data_admissão, cidade, escolaridade, grau, cursos_realizados, experiências_profissionais, atribuições_desenvolvidas, setor_que_trabalhou, data_entrada, data_saída, motivo_saída, atribuições_que_gostaria, horário, setores/divisões/diretorias_gostaria, motivos_descontente, observações, data_preenchimento, questão1, questão2, questão3, questão4, questão5, questão6, questão7, resultado, justificativa, data_resultado).

4.6 Aplicação da 1ª Forma Normal - 1FN

A 1FN diz que todos os atributos devem conter um valor atômico (único), ou seja, os dados inclusos dentro do campo não devem se repetir. Ex.: o campo que representa os cursos feitos pelo requerente será preenchido quantas vezes no formulário? Depende de quantos cursos ele fez: na possibilidade de ser preenchido duas vezes, o campo é retirado para uma tabela associativa.

Remanejamento (código, nome, sexo, data_nascimento, categoria_funcional, lotação_atual, data_admissão, cidade, data_preenchimento, questão1, questão2, questão3, questão4, questão5, questão6, questão7, resultado, justificativa, data_resultado);

Permaneceram na tabela remanejamento apenas os campos com valores únicos, os campos que podem ser preenchidos uma ou mais vezes são retirados para as tabelas associativas:

- Remanejamento/escolaridade (**código_remanejamento**, escolaridade, grau);
- Remanejamento/Cursos (**código_remanejamento**, cursos);
- Remanejamento/experiências profissionais (**código_remanejamento**, experiências_profissionais);
- Remanejamento/atribuições desenvolvidas (**código_remanejamento**, atribuições);
- Remanejamento/setores que trabalhou (**código_remanejamento**, setor, data_entrada, data_saída, motivo_saída);
- Remanejamento/atribuições que gostaria (**código_remanejamento**, atribuições, horário);
- Remanejamento/setores-divisões-diretorias que gostaria (**código_remanejamento**, setor);
- Remanejamento/motivos descontentamento (**código_remanejamento**, motivos, observações).

4.7 Aplicação da 2ª Forma Normal - 2FN

A 2FN diz que todos os atributos que não forem chave têm que ser dependentes diretos da chave, ou seja, os atributos que estão contidos no requerimento têm que depender diretamente da chave código requerimento.

Inicialmente, observa-se a entidade que possui chave primária concatenada. Para aquelas que satisfazem esta condição, analisar se existe algum atributo ou conjunto de atributos com dependência direta à chave primária.

Remanejamento (código, código_funcionário, categoria_funcional, lotação_atual, data_admissão, cidade, data_preenchimento, questão1, questão2, questão3, questão4, questão5, questão6, questão7, resultado, justificativa, data_resultado);

- Verifique que o campo categoria funcional, lotação atual e cidade devem permanecer na entidade Remanejamento, tendo em vista que, se, por acaso, se desejar buscar um formulário emitido há cinco anos, cujo funcionário alterou sua categoria funcional, lotação atual ou cidade, os dados que deverão aparecer no formulário preenchido há cinco anos devem ser o correspondente à categoria funcional, lotação atual ou cidade, preenchido antigamente, e não os dados novos.

- Funcionário (**código**, nome, sexo, data_nascimento);

- Remanejamento/escolaridade (**código_remanejamento_escolaridade**, grau);

- Remanejamento/Cursos (**código_remanejamento_cursos**);

- Remanejamento/experiências profissionais (**código_remanejamento_experiências**);

- Remanejamento/atribuições desenvolvidas (**código_remanejamento_atribuições**);

- Remanejamento/setores que trabalhou (**código_remanejamento_setor**, data_entrada, data_saída, motivo_saída);

- Remanejamento/atribuições que gostaria (**código_remanejamento_atribuições**, horário);

- Remanejamento/setores-divisões-diretorias que gostaria (**código_remanejamento_setor**);

- Remanejamento/motivos descontentamento (**código_remanejamento_motivos**, observações);

Aplicação da 3ª Forma Normal – 3FN

A 3FN diz que o atributo precisa estar na segunda forma normal, e todos os atributos que não são chave, não podem depender de outros atributos que também não são chave.

Uma Entidade está na 3FN se nenhum de seus atributos possui dependência em relação a outros atributos da entidade. Ao verificar a existência de dependência entre as Entidades e seus atributos, devemos criar outra entidade que contenha os atributos dependentes.

Remanejamento (código, código_funcionário, código_categoria_funcional, código_lotação, data_admissão, código_cidade, data_preenchimento, questão1, questão2, questão3, questão4, questão5, questão6, questão7, resultado, justificativa, data_resultado);

- Categoria Funcional **(código**, descrição);
- Lotação (**código**, descrição); Cidade (código, descrição);
- Funcionário (**código**, nome, sexo, data_nascimento);
- Remanejamento/escolaridade (**código_remanejamento, código_escolaridade**, grau);
- Escolaridade (**código**, descrição);
- Remanejamento/Cursos (**código_remanejamento, código_cursos**);
- Cursos (**código**, descrição);
- Remanejamento/experiências profissionais (**código_remanejamento, código_experiências**);
- Experiências Profissionais (**código**, descrição);
- Remanejamento/atribuições desenvolvidas (**código_remanejamento, código_atribuições**);
- Atribuições (**código**, descrição);
- Remanejamento/setores que trabalhou (**código_remanejamento, código_setor**, data_entrada, data_saída, motivo_saída);
- Setores (**código**, descrição);
- Remanejamento/atribuições que gostaria (**código_remanejamento, código_atribuições**, horário);
- Remanejamento/setores-divisões-diretorias que gostaria (**código_remanejamento, código_setor**);
- Remanejamento/motivos descontentamento (**código_remanejamento, código_motivos**, observações);
- Motivo descontentamento (**código**, descrição).

5 Referencias

Este material foi baseado em:

FRANCO, Matheus. **Sistemas de Gerenciamento de Banco de Dados**. São João da Boa Vista: Instituto Federal de Educação, Ciência e Tecnologia de São Paulo/Rede e-Tec, 2013.

MACHADO, Felipe N. R. **Projeto de Banco de Dados**: Uma visão prática. São Paulo: Érica, 2004