**Fundamentos de Banco de dados – IFRS - Dia 1 - 25/08/2025**

#### **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:

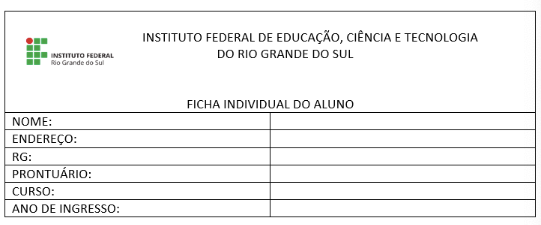


Figura 1 - Exemplo de Ficha de Matrícula

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 ocorra 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.

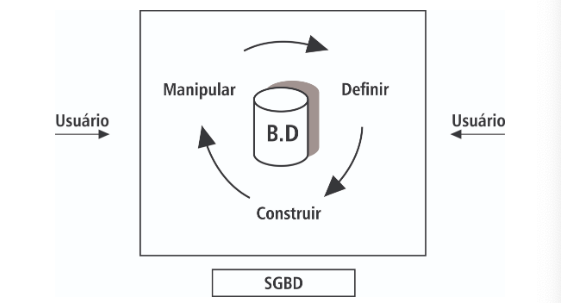


Figura 2 - Sistema Gerenciador de Banco de Dados (SGBD)

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.

**Dia 2 - 25/08/2025**

**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 necessidade 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:

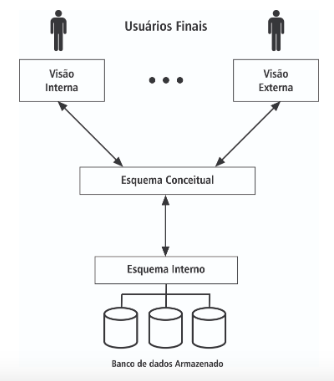
* Nível externo: possui as diversas descrições do BD de acordo com os grupos de usuários;
* Nível conceitual: descreve a estrutura de todo o BD para uma determinada comunidade de usuários, ocultando detalhes sobre a organização física dos dados e apresentando a descrição lógica dos dados e das ligações 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 1 - 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 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**

https://youtu.be/qgnuH\_qSI9o

**1.7 Vídeo complementar 2: Conceitos fundamentais sobre bancos de dados**

<https://youtu.be/tEEAIs6aB2s>

**Resumo do Day 1**

1.1 Introdução

No cotidiano, precisamos armazenar e recuperar informações, como listas de endereços, dados financeiros e outros registros. Para serem úteis, esses dados devem ser seguros, confiáveis e facilmente acessíveis. Dados armazenados sem um método eficiente de recuperação tornam-se praticamente inúteis. Além disso, as informações geralmente não estão isoladas, mas sim relacionadas, como fichas de matrícula de alunos com diversos atributos.

Os usuários necessitam realizar operações sobre essas informações, como adicionar, consultar, atualizar ou excluir dados. O advento da tecnologia trouxe os Bancos de Dados (BD) como solução, definidos como coleções logicamente coerentes de dados com significado inerente. Um BD é projetado para um propósito específico, refletindo alterações no mundo real (denominado "minimundo"). Sistemas computacionais gerenciam bancos de dados através de programas chamados Sistemas de Gerenciamento de Bancos de Dados (SGBD). O objetivo principal de um SGBD é oferecer um ambiente eficiente para armazenamento e recuperação de informações.

1.2 História

Os primeiros sistemas de bases de dados surgiram no final da década de 1960, inicialmente simples, mas evoluíram rapidamente devido às necessidades das empresas. Pioneiras como IBM, Oracle e Sybase impulsionaram esse campo. Em 1970, a IBM publicou trabalhos sobre bancos de dados relacionais, propondo o uso de cálculo e álgebra relacional para facilitar a manipulação de dados, resultando no desenvolvimento da Structured Query Language (SQL), que se tornou o padrão da indústria para bancos de dados relacionais.

Apesar da IBM ter concebido o conceito de SQL, o primeiro sistema comercial foi lançado pela Honeywell em 1976. O trabalho da IBM evoluiu para sistemas como SQL/DS e DB2, estabelecendo a base dos bancos de dados modernos.

1.3 Usuários de Bancos de Dados

Os usuários de bancos de dados variam em responsabilidades e envolvimento com os dados, sendo divididos em:

* Administradores de Bancos de Dados (DBA): Supervisionam o uso do BD, gerenciam segurança, desempenho e autorizam acessos. Em grandes organizações, pode haver uma equipe de DBAs.
* Analistas de Bancos de Dados (Projetistas): Identificam os dados a serem armazenados, escolhem estruturas adequadas e integram diferentes visões dos usuários para criar o projeto do BD.
* Usuários Finais: Acessam o BD para consultar, modificar e gerar relatórios. Podem ser usuários comuns ou avançados, com diferentes níveis de interação.
* Analistas de Sistemas e Programadores de Aplicações: Desenvolvem especificações para atender às necessidades dos usuários finais e implementam-nas como programas.

1.4 Níveis de Abstração

Um banco de dados possui três níveis de abstração conforme a arquitetura ANSI/SPARC:

1. Nível Externo: Representa as visões dos dados pelos diferentes grupos de usuários.
2. Nível Conceitual: Descreve a estrutura lógica dos dados, ocultando detalhes físicos.
3. Nível Interno: Trata da estrutura física de armazenamento e caminhos de acesso aos dados.

Esses níveis permitem converter representações entre si através de mapeamentos.

1.5 Características de um SGBD

Um Sistema de Gerenciamento de Bancos de Dados apresenta as seguintes propriedades:

* Controle de Redundância: Minimiza a duplicação de dados para evitar inconsistências, reduzir o uso de espaço e aumentar a eficiência.
* Compartilhamento de Dados: Garante o controle de concorrência para que múltiplos usuários possam acessar os dados de forma consistente e correta.
* Controle de Acesso: Permite definir autorizações específicas para diferentes usuários, restringindo ou ampliando permissões de acordo com o nível de acesso necessário.
* Representação de Dados Abstrata: Oferece representações em níveis abstratos (externo, conceitual e interno) para diferentes propósitos.

O SGBD combina o banco de dados e seu software, formando um Sistema de Bancos de Dados (SBD). Ele organiza os dados seguindo regras que maximizam a eficiência do armazenamento e acesso.

Conclusão

Bancos de dados são essenciais para gerenciar informações de maneira eficiente e estruturada. Desde sua criação, os SGBDs têm evoluído para atender às demandas de organizações, oferecendo recursos avançados de controle, compartilhamento e abstração de dados.

Questionário com Respostas

1. O que é um Banco de Dados (BD)?

* Resposta: É uma coleção logicamente coerente de dados com significado inerente, projetada para armazenar informações relacionadas e refletir alterações no mundo real.

2. Quais são os principais objetivos de um Sistema de Gerenciamento de Bancos de Dados (SGBD)?

* Resposta: Fornecer um ambiente eficiente para armazenar, recuperar e gerenciar informações, controlando redundâncias, compartilhando dados, gerenciando acessos e abstraindo representações.

3. Quais são os níveis de abstração de um banco de dados?

* Resposta:
  1. Nível Externo: Visão dos usuários sobre os dados.
  2. Nível Conceitual: Estrutura lógica do banco de dados.
  3. Nível Interno: Organização física e caminhos de acesso.

4. Quem são os usuários de um banco de dados e quais suas funções?

* Resposta:
  + DBA (Administrador): Gerencia o banco, controla acesso e desempenho.
  + Analista de Bancos de Dados: Projeta e estrutura os dados.
  + Usuários Finais: Consultam, modificam e geram relatórios.
  + Analistas e Programadores: Desenvolvem e implementam transações.

5. Qual foi o papel da IBM na história dos bancos de dados?

* Resposta: A IBM desenvolveu os fundamentos dos bancos relacionais e a linguagem SQL, além de liderar pesquisas que resultaram no Sistema R, precursor do DB2.

6. Quais problemas podem surgir devido à redundância de dados?

* Resposta:
  + Inconsistências nos dados.
  + Aumento do uso de espaço de armazenamento.
  + Maior esforço computacional para atualizações.

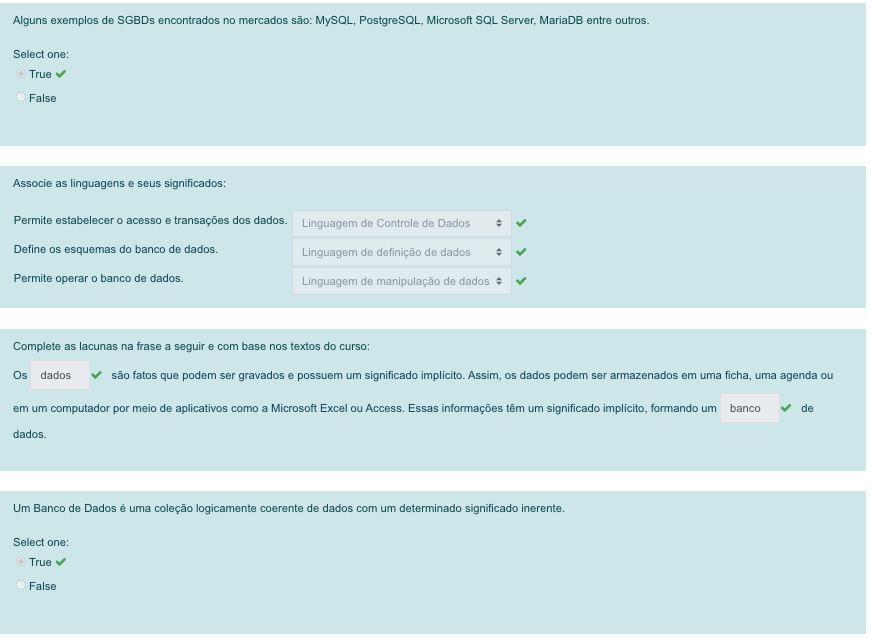
7. O que é o minimundo em relação aos bancos de dados?

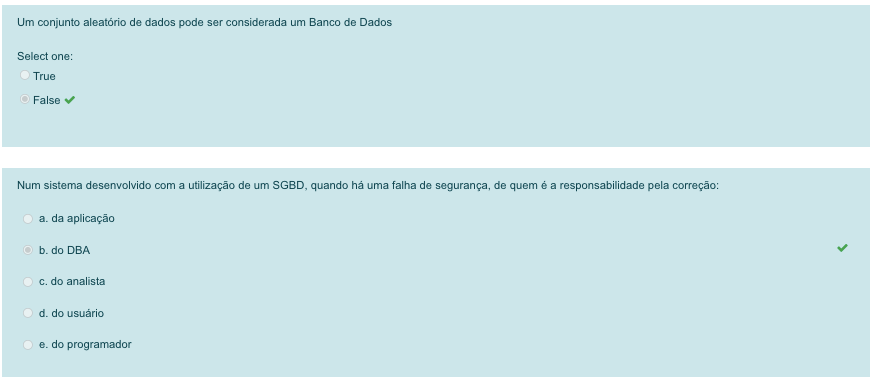
* Resposta: É a parte restrita do mundo real que o banco de dados representa, refletindo suas alterações.

8. Quais são as vantagens de um SGBD em ambientes multiusuários?

* Resposta:
  + Controle de concorrência.
  + Compartilhamento consistente de dados.
  + Controle de acesso personalizado para usuários.

1.8 Teste seus conhecimentos







#### **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.

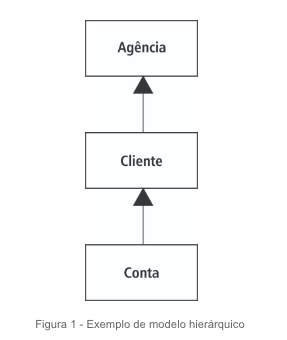


Figura 1 - Exemplo de modelo hierárquico

Descrição da imagem: Há palavras dentro de retângulos. Na parte inferior, a primeira palavra é Conta, quer liga-se à Cliente, logo acima, que liga-se à Agência, no topo, representando um 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:

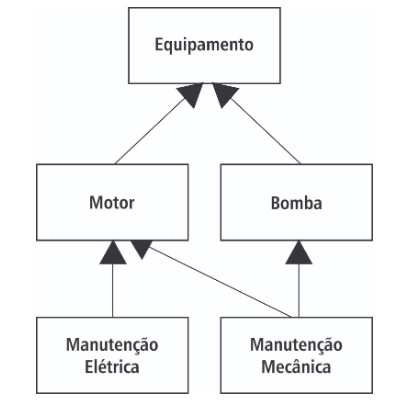


Figura 1 - Exemplo de modelo em rede

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 à 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; até chegar ao
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:

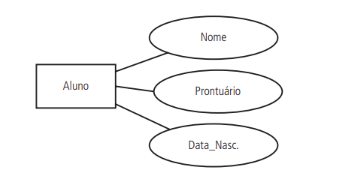


Figura 1 - Entidade com seus atributos

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.

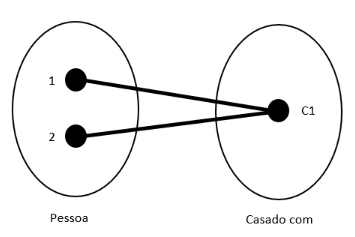


Figura 1 - Exemplo de relacionamento unário

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

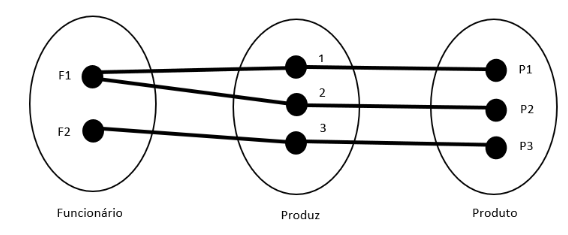


Figura 2 - Exemplo de relacionamento binário

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

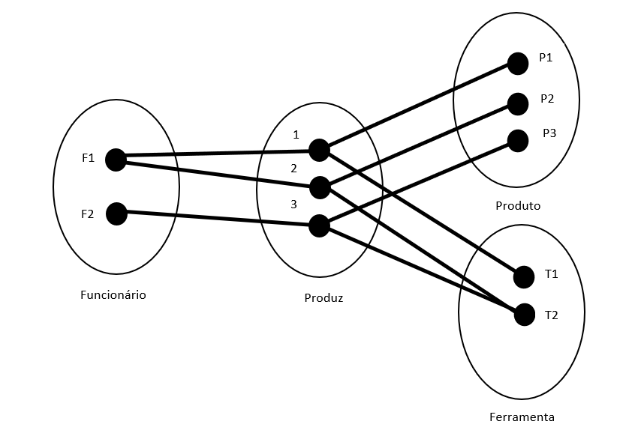


Figura 3 - Exemplo de Relacionamento ternário

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

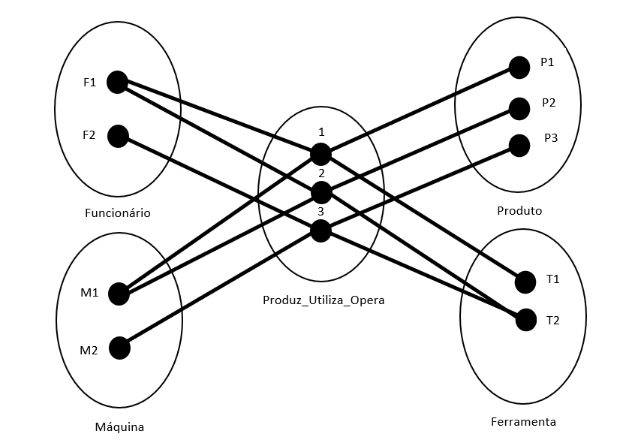


Figura 4 - Exemplo de relacionamento quaternário

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

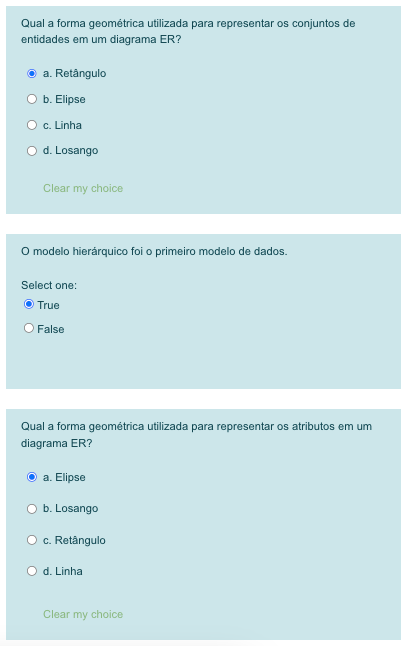
Os relacionamentos entre entidades possuem certas restrições que limitam as combinações possíveis das entidades que dele participam. Uma destas restrições é a razão de **cardinalidade**. A quantidade de Entidades envolvidas em cada Relacionamento é determinada pela Cardinalidade do Tipo de Relacionamento,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

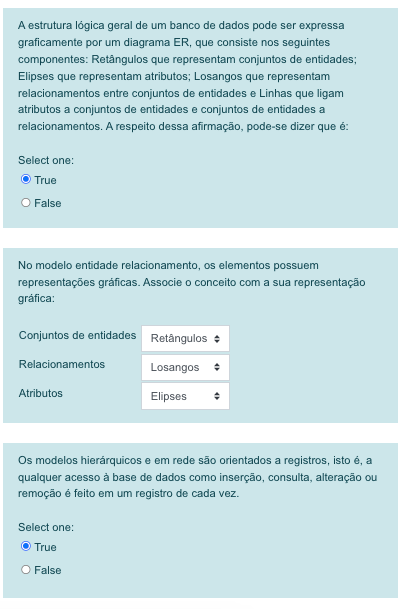
Até agora vimos, de modo geral, como construir um diagrama de entidade relacionamento (DER) para projetar um banco de dados. Aprendemos os quatro principais construtores deste modelo: os conjuntos de entidades (CE), os conjuntos de relacionamentos (CR), os atributos de entidades e relacionamento. Agora, 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.

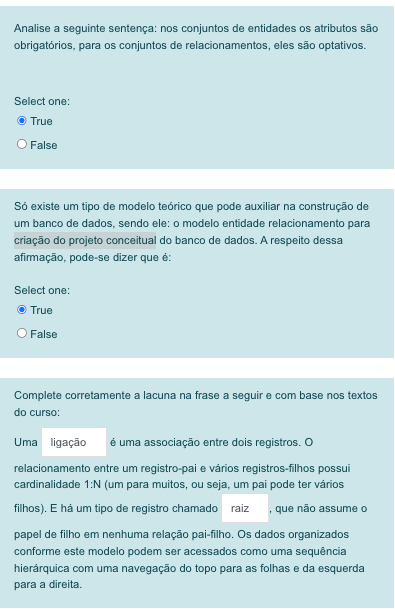
## 2.2.2 Link: Software DIA

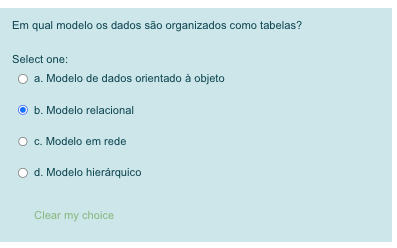
2.2. Link: Software DIA

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









Parte superior do formulário

#### **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 **Número** 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

#### 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:

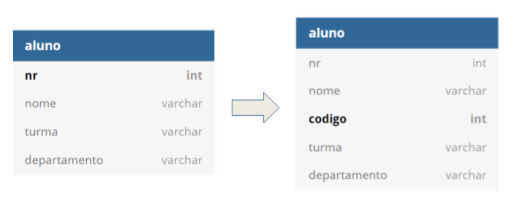


Figura 1 - Representação da tabela "aluno"

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 2 - 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 3 - 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 4 - 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.

**Resumo do Day 2**

#### 1. Modelos de Dados

Os Sistemas Gerenciadores de Bancos de Dados (SGBDs) surgiram para resolver limitações dos sistemas baseados em arquivos. Eles adotam diferentes modelos de dados, que determinam como as informações são organizadas, acessadas e manipuladas. Os principais modelos incluem:

##### 1.1 Modelo Hierárquico

* Estrutura em forma de árvore, com registros organizados em hierarquias de pai-filho.
* Cada nó (registro) tem:
  + Campos (atributos) que armazenam informações.
  + Relacionamentos com outros registros (pai-filho), sempre na proporção 1:N (um pai pode ter vários filhos).
* Características principais:
  + Dados são acessados na ordem hierárquica, percorrendo a árvore do topo às folhas.
  + Problemas: replicação de dados (causa inconsistência) e desperdício de espaço.
* Exemplo: Uma agência bancária como pai pode ter vários clientes como filhos; os clientes podem ter várias contas.

##### 1.2 Modelo em Rede

* Extensão do modelo hierárquico que elimina a hierarquia rígida.
* Registros são organizados como grafos, permitindo:
  + Relacionamentos 1:N e também N:N.
  + Acesso direto a qualquer registro, sem a necessidade de começar pela raiz.
* Vantagem: Modelagem mais próxima da realidade, com menos restrições.
* Desvantagem: Substituído pelo modelo relacional devido à sua complexidade.
* Exemplo: Equipamentos com várias bombas e motores, todos podendo sofrer manutenção mecânica ou elétrica.

##### 1.3 Modelo Orientado a Objetos

* Representa os dados como coleções de objetos, similares aos utilizados em programação orientada a objetos.
* Objetos possuem:
  + Propriedades (atributos) que descrevem suas características.
  + Métodos (operações) que definem seu comportamento.
* Aplicado mais como um modelo conceitual do que em implementações práticas.

#### 2. Modelagem de Dados

A modelagem de dados é o processo de criar representações conceituais da estrutura de um banco de dados. O Modelo Entidade-Relacionamento (MER) é amplamente usado por sua proximidade com a visão do usuário.

##### 2.1 Componentes do MER

* Conjunto de Entidades: Representam objetos no mundo real (ex.: cliente de um banco).
* Conjunto de Atributos: Definem características das entidades. Por exemplo, atributos de um cliente podem ser "nome" e "CPF".
* Conjunto de Relacionamentos: Associações entre entidades. Um exemplo é a relação entre clientes e contas bancárias.
* Cardinalidade: Determina o número mínimo e máximo de entidades relacionadas. Por exemplo:
  + 0:N: Um cliente pode ter nenhuma ou várias contas.
  + 1:1: Um departamento pode ter apenas um gerente.

##### 2.2 Diagramas Entidade-Relacionamento (DER)

* Usados para representar graficamente o MER.
* Elementos:
  + Retângulos: Conjuntos de entidades.
  + Elipses: Atributos.
  + Losangos: Relacionamentos.
  + Linhas: Conexões entre os elementos.

##### 2.3 Processo de Modelagem

1. Coleta de requisitos: Identificar as necessidades do banco de dados.
2. Modelo conceitual (MER): Criar o diagrama de entidades e relacionamentos.
3. Modelo lógico (relacional): Converter o diagrama em tabelas.
4. Implementação no SGBD: Inserir as tabelas no sistema, como o MySQL.

#### 3. Modelo Relacional

O modelo relacional é o mais amplamente utilizado, devido à sua simplicidade e flexibilidade. Ele representa os dados como tabelas bidimensionais, nas quais:

* Linhas (tuplas): Representam instâncias de dados.
* Colunas (atributos): Definem as características de cada entidade.
* Esquema: Estrutura das tabelas, composta por seus nomes e atributos.

##### 3.1 Elementos do Modelo Relacional

* Atributos-chave: Identificam unicamente cada linha (ex.: CPF em uma tabela de clientes).
  + Chave primária: Atributo ou conjunto de atributos escolhido para ser o identificador principal.
  + Chave estrangeira: Relaciona uma tabela com outra, garantindo integridade referencial.
* Domínios: Definem os valores possíveis para cada atributo (ex.: a idade deve ser um número inteiro entre 0 e 150).

##### 3.2 Representação Relacional

* Tabelas são criadas a partir do MER, seguindo regras de mapeamento.
* Exemplo: Um relacionamento entre "Funcionário" e "Setor" pode ser representado em duas tabelas conectadas por chaves primárias e estrangeiras.

##### 3.3 Vantagens do Modelo Relacional

* Flexibilidade: Permite manipular dados sem caminhos pré-definidos.
* Redução de Redundância: Uso de normalização para evitar duplicações.
* Facilidade de Uso: Dados organizados como tabelas, formato intuitivo para usuários.

#### 4. Normalização

A normalização é o processo de organizar dados em tabelas para:

* Evitar redundâncias.
* Garantir integridade dos dados.
* Facilitar a manutenção do banco.

#### 5. Conclusão

A escolha do modelo e da modelagem é fundamental para o sucesso de um banco de dados, garantindo eficiência, segurança e consistência. O modelo relacional se consolidou como padrão devido à sua robustez e facilidade de uso.