

**BandTec**  
**DIGITAL SCHOOL**



## **BD – Banco de Dados**

Aula inaugural

© Profa. Célia Taniwaki

# Apresentação da Professora

- Ciência da Computação – IME/USP
- Administração de Empresas – FGV
- Analista de Software – Itautec Informática S.A.
  - Interpretador das linguagens BASIC e LOGO para microcomputadores de 8 e 16 bits
  - Monitor transacional GRIP – middleware para desenvolvimento de aplicações de automação bancária e comercial – Plataformas: DOS, NetWare, Windows, Linux
- Mestrado em Engenharia Elétrica – POLI/USP
  - Área: Inteligência Artificial (Linguagem Natural)

# Apresentação da Professora (Cont.)

- Tradutora técnica de artigos de revistas e de livros de informática
- Analista de Software – CIS Eletrônica
  - Software embarcado de equipamentos que fazem captura de dados: leitor de cartão magnético, leitor de código de barra, leitor de cheque, leitor biométrico
  - Software de drivers desses dispositivos
- Professora universitária – desde 2005
  - UNIB, Faculdade Drummond, Unip, Uninove, ESEG
  - BandTec – desde 2011
    - Disciplinas: Sistemas Operacionais, Algoritmos, Paradigmas de Programação, Programação Orientada a Objetos, Estrutura de Dados, Sistemas Distribuídos, Arquitetura Computacional, Banco de Dados

# O que é Banco de Dados? O que são Dados?

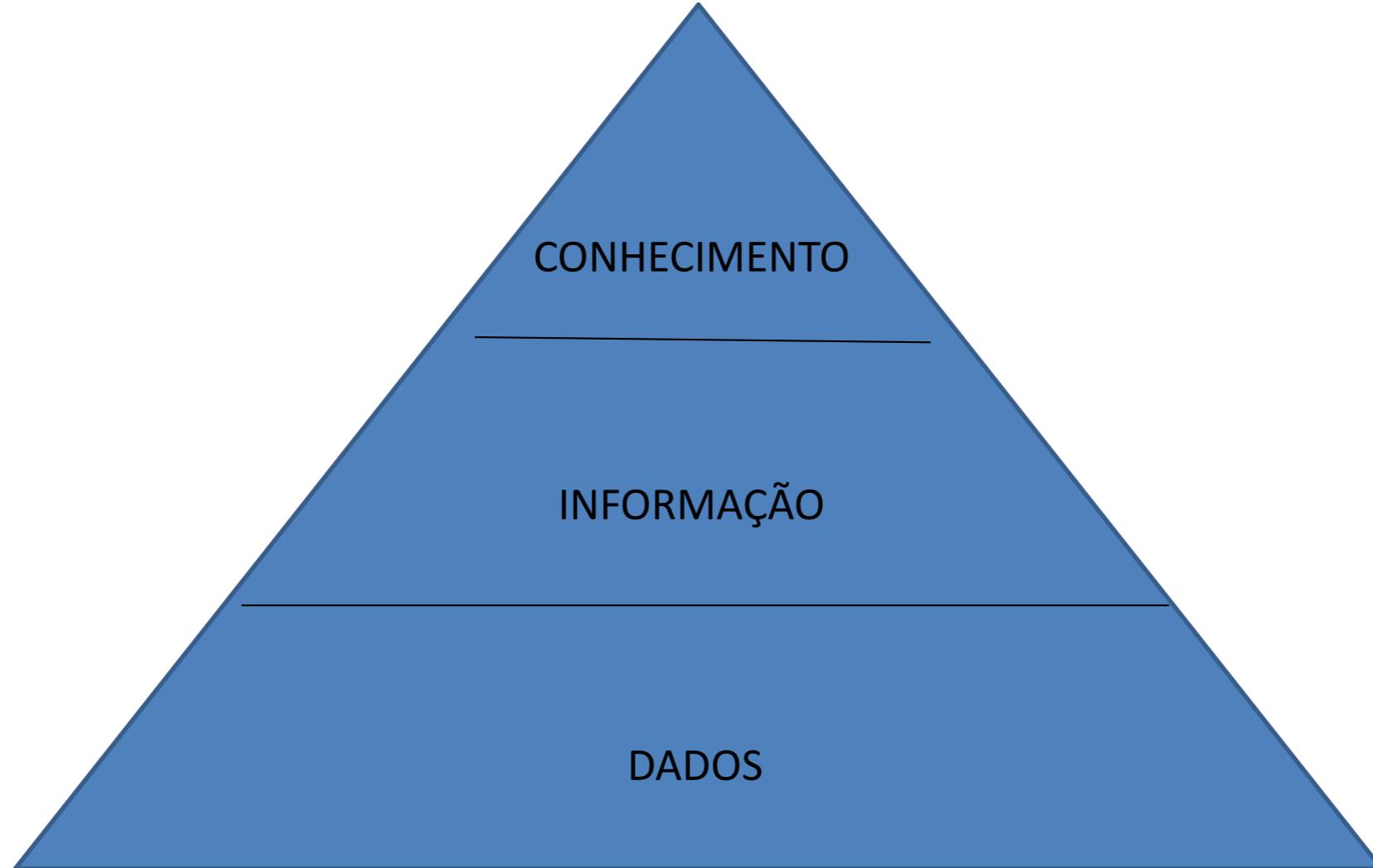
??!!!!???????



# O que são dados? O que são informações?

- Somos da área de TI – Tecnologia da Informação
- O que vem a ser informação?
- Qual a diferença entre dados e informações?
- Qual a diferença entre dados, informações e conhecimento?
- Pesquisa em grupo

# Dados vs Informações vs Conhecimento



# Onde armazenamos os dados?

- Na memória do computador?
- No pendrive? No disco rígido?
- Vamos armazenar os dados dos colegas em arquivos?

# Ementa

- Conceitos de bancos de dados. Modelo Relacional: MER e DER – Modelo e Diagrama Entidade Relacionamento. Modelos Conceitual, Lógico e Físico de Dados. Transformação de modelos físicos. Normalização. Tipos de banco de dados. Fundamentos de bancos de dados relacionais. Sistemas Gerenciadores de Bancos de dados Relacionais. SQL. Consultas.

# Conteúdo Programático

- Conceitos básicos de banco de dados.
- Conceitos de modelagem de dados conceitual. Modelo Entidade Relacionamento.
- Conceitos de modelo lógico.
- Normalização (1FN, 2FN, 3FN).
- Modelo físico. Dicionário de dados.
- SQL.
- Criação de banco de dados. Consultas a banco de dados.

# Objetivos

- Ao final do semestre, o aluno estará apto a:
  - Modelar dados
  - Utilizar SQL para
    - Criar banco de dados
    - Inserir dados no banco de dados
    - Realizar operações no banco de dados (atualizar, consultar, excluir, inserir, ...)

# Banco de Dados – Nosso caminho



introdução

- Conceitos Básicos
- Comandos DDL
- Comandos DML
- Administração de BD



14/09

- Modelagem Conceitual
- Modelagem Lógica
- Relacionamento 1–N
- Restrições



26/10

- Modelagem Física
- Relacionamento N–N
- Normalização de Dados

07/12



Final de Semestre



- Apresentação PI
- Avaliação Integrada

**LEGENDA**

- Conteúdo ✓ Conteúdo Finalizado
- Entregável PI ✓ Entregável Finalizado



Onde Estamos



- Semana final das Sprints
- Semana das Entregas de PI

# Avaliações

- Avaliações Continuadas 1, 2 e 3 (uma para cada Sprint)
  - Exercícios e provinhas ao longo do semestre
  - Trabalho individual - Avaliação continuada
- Avaliação Integrada – no final do semestre
- Média final = Média das ACs \* 0,4 + Nota da AI \* 0,6
- Avaliação Substitutiva:
  - Caso a média final seja < 6,0 ou a nota da AI < 6,0
  - SUB substitui a nota da AI, se ela for maior

# Regras a serem observadas!

- Respeitar o horário das aulas
  - Não atrasar para o início: 10:15 hrs
  - Não atrasar após o intervalo
    - 20 minutos com tolerância de 10 min
  - Nem sair antes do final
    - Chamada no final da aula
  - Aproveitar o horário da aula o máximo possível!
  - Cuidado com distrações (Internet, celular, conversas paralelas)

# Regras a serem observadas!

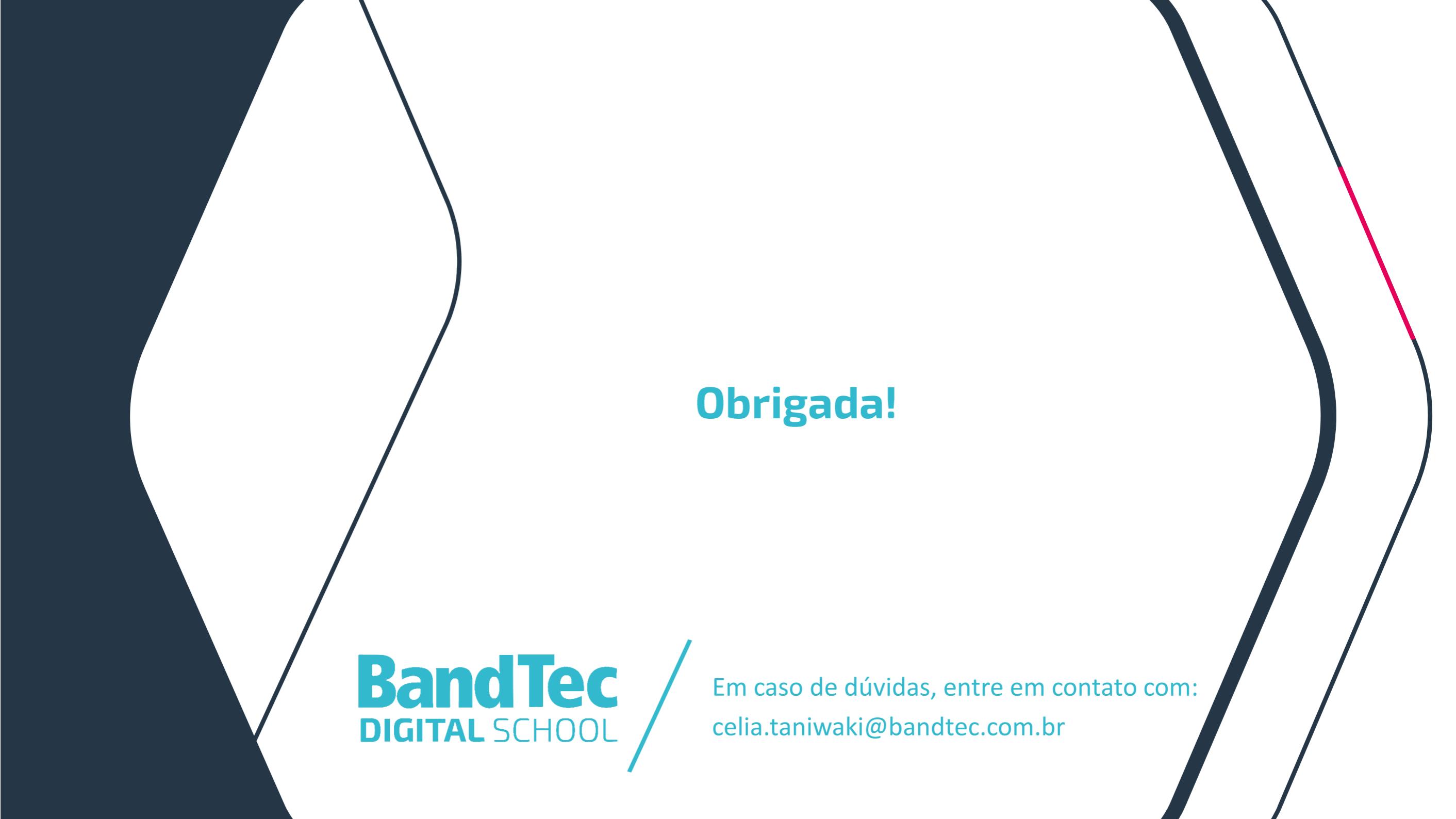
- Para aprender, é fundamental:
  - Praticar, praticar, praticar.....
- Procure praticar os exercícios propostos
  - Durante a aula e também semanalmente, fora do horário da aula!
- Lembre-se:
  - Copiar do colega ou da Internet não é aprender!
  - Aprendemos quando tentamos, nós mesmos, resolver o problema!
- Não fique com dúvidas!
  - Procure resolver suas dúvidas o quanto antes!
  - Em sala de aula, ou por e-mail

# Referências Bibliográficas

- PUGA, Sandra; FRANÇA, Edson; GOYA, Milton. **Banco de Dados: Implementação em SQL, PL/SQL e Oracle 11g.** São Paulo: Pearson Education do Brasil, 2013. 329 p.
- MACHADO, Felipe Nery Rodrigues. **Banco de dados: projeto e implementação.** São Paulo: Érica, 2004. 398 p.
- SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de banco de dados.** Tradução de Daniel Vieira. Revisão técnica Luis Ricardo de Figueiredo; Caetano Traina Jr. 3. ed. São Paulo: Pearson Makron Books, 2007. 778 p.

## Referências Bibliográficas

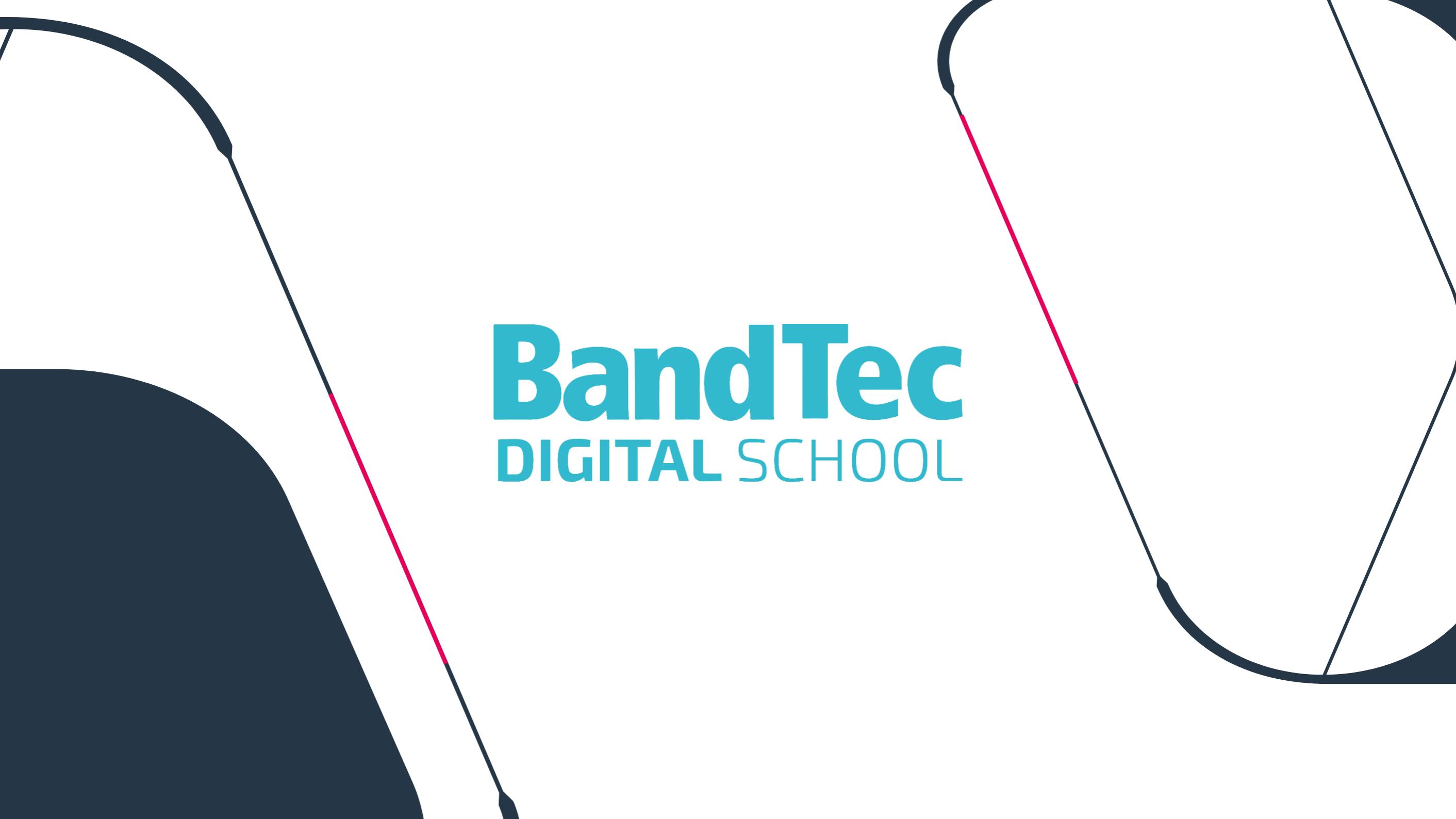
- HEUSER, Carlos A. **Projeto de banco de dados.** 6. ed. Porto Alegre: Bookman, 2009.
- COUGO, Paulo. **Modelagem conceitual e projeto de banco de dados.** Rio de Janeiro: Elsevier, 1997. 284 p.
- ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de banco de dados.** Tradução de Marília Guimarães Pinheiro et al. 4. ed. São Paulo: Pearson Addison Wesley, 2005. 724 p.
- MACHADO, Felipe Nery Rodrigues; ABREU, Maurício Pereira de. **Projeto de banco de dados: uma visão prática.** 15 ed. São Paulo: Érica, 2007. 300 p.



Obrigada!

**BandTec**  
DIGITAL SCHOOL

Em caso de dúvidas, entre em contato com:  
[celia.taniwaki@bandtec.com.br](mailto:celia.taniwaki@bandtec.com.br)



**BandTec**  
**DIGITAL SCHOOL**



# **BD – Banco de Dados**

Aula01 - Conceitos

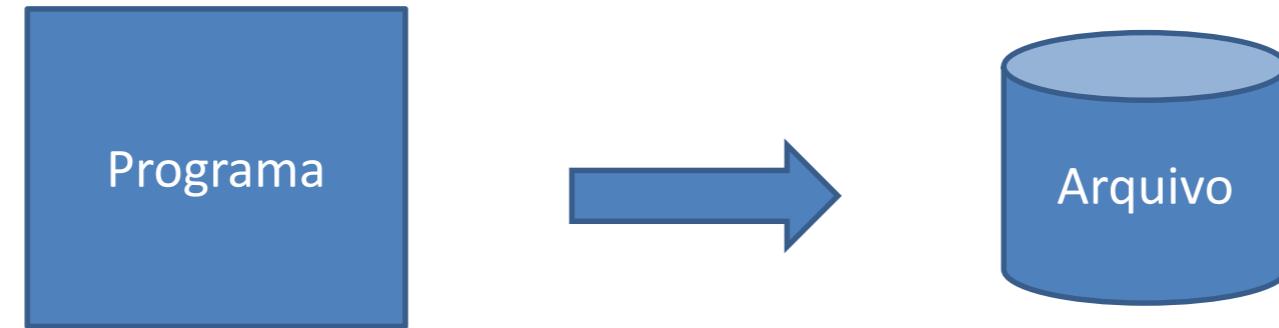
© Profa. Célia Taniwaki

# Importância dos Bancos de Dados

- Competitividade das empresas ⇒ depende de dados precisos e atualizados
- Crescimento da empresa ⇒ aumenta a dependência por dados abundantes e complexos
- Surge a necessidade de ferramentas de gerenciamento, extração rápida e precisa de informações
- Solução: Sistema Gerenciador de Banco de Dados (SGBD).

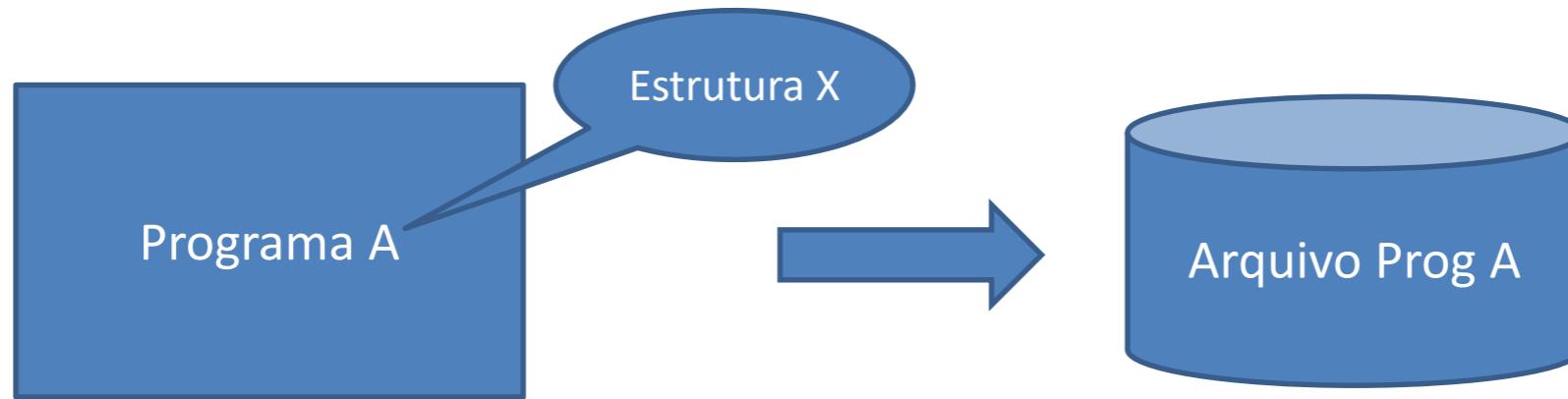
# Historicamente...

- Primeiros programas de computador ⇒ objetivo de armazenar e manipular dados
- Programas gravavam seus dados em disco, seguindo estruturas próprias.

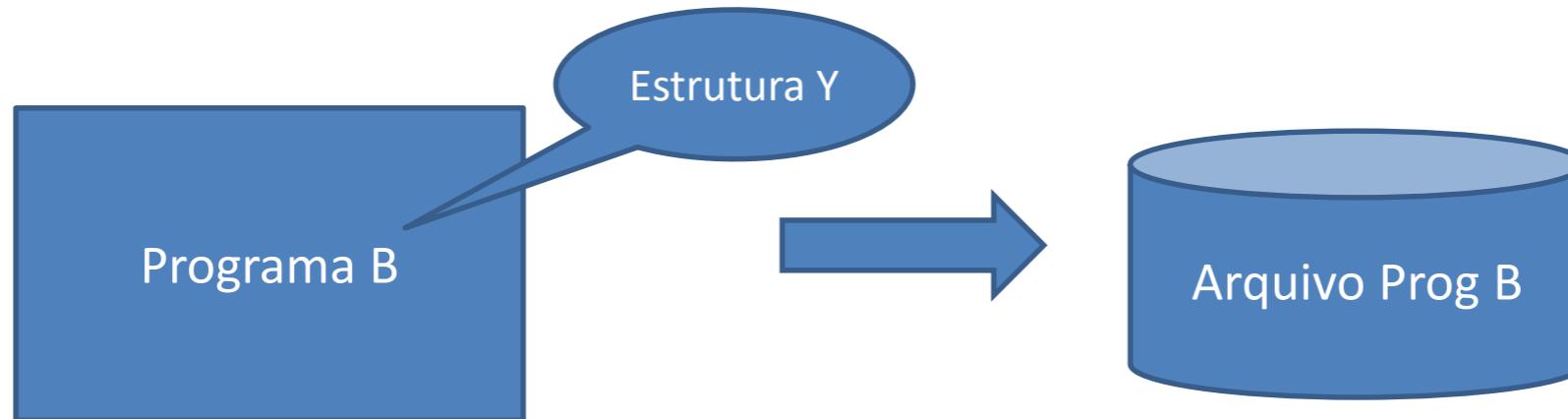


# Historicamente...

- Programa A gravava seus dados usando uma estrutura X.

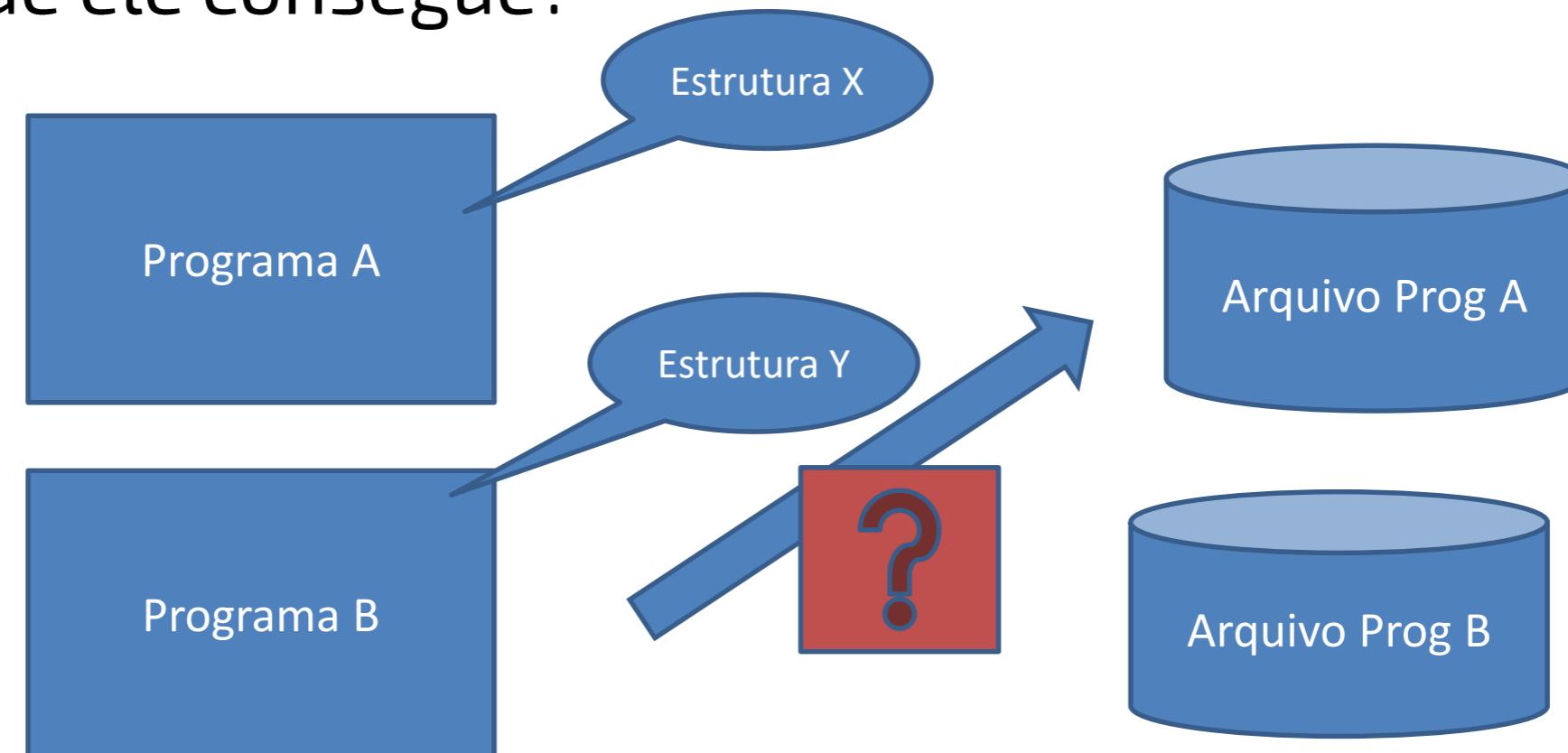


- Programa B gravava seus dados usando uma estrutura Y.



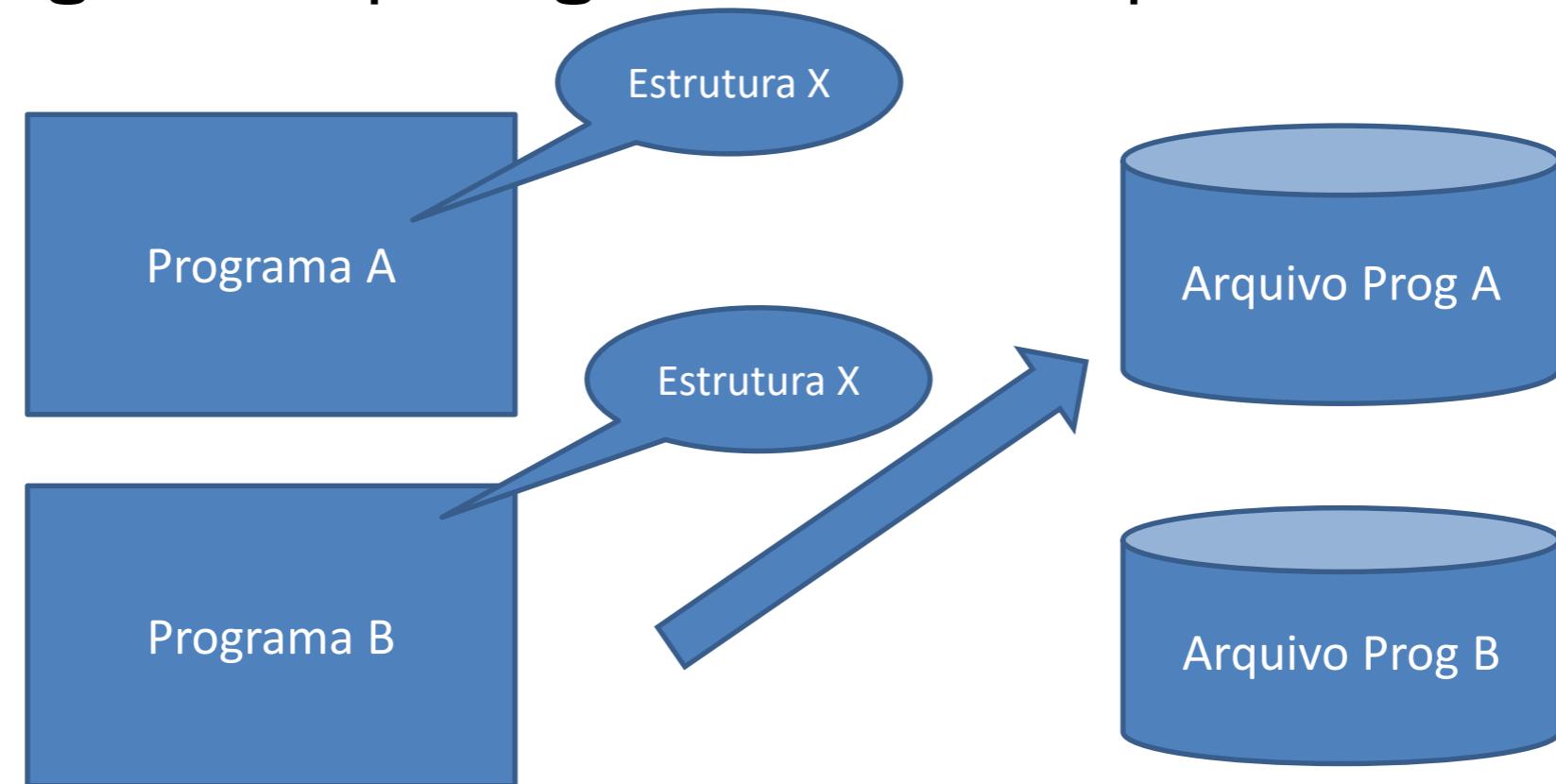
# Historicamente...

- Vamos supor que o programa B queira acessar o arquivo do programa A.
- Será que ele consegue?



# Historicamente...

- Para que o programa B consiga acessar o arquivo do programa A, ele precisa conhecer a estrutura X usada pelo programa A para gravar o seu arquivo.



# Historicamente...

- Programas que não conheciam a estrutura dos arquivos dos outros programas não podiam utilizar os dados.
- Se vários programas precisassem acessar os dados de um mesmo arquivo, todos os programas teriam que conhecer e manipular as mesmas estruturas.

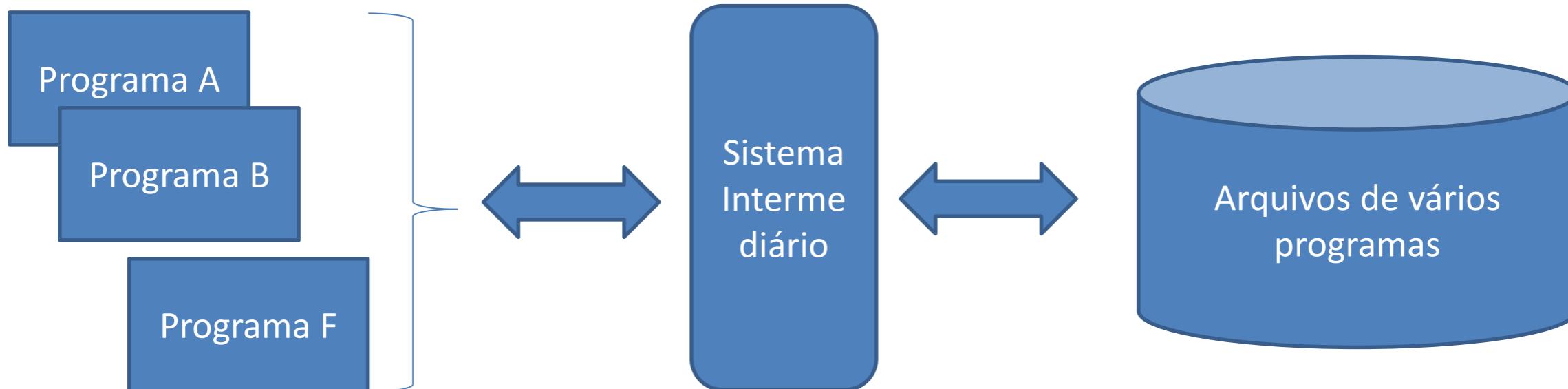


# Historicamente...

- Vamos supor que os programas da figura anterior foram alterados para conhecer as estruturas dos dados dos demais programas.
- Dessa forma, os arquivos puderam ser compartilhados entre os programas.
- Mas, e se o programa A tivesse que realizar uma mudança na estrutura dos arquivos do programa A ???
  - Todos os programas que acessam esse mesmo arquivo deveriam ser alterados
- Grande problema:
  - Garantir a unicidade das estruturas de dados entre os diversos programas

# Historicamente...

- Para evitar esse problema, providenciaram um sistema intermediário, que:
  - Conhece a estrutura de dados do arquivo.
  - Fornece apenas os dados que cada programa precisa.
  - Armazena adequadamente os dados de cada programa.

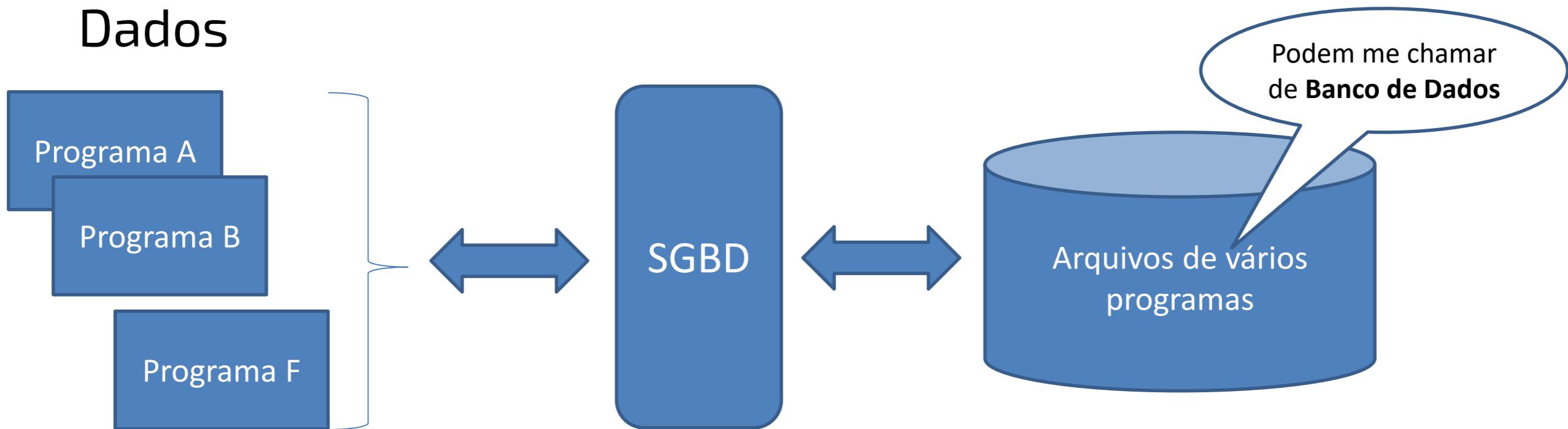


# Historicamente...

- Assim, utilizando esse sistema intermediário:
  - Programas podem obter apenas os dados que lhes interessam
  - Programas não precisam conhecer os detalhes de como os dados estão gravados fisicamente
  - Programas não precisam ser modificados se a estrutura de dados que utilizam não for modificada
  - Alterações ficam concentradas nesse sistema intermediário

# Historicamente...

- Com o tempo, esse sistema intermediário passou a gerenciar vários arquivos
- Coleção de arquivos: Banco de Dados
- Sistema intermediário: Sistema Gerenciador de Banco de Dados



# Historicamente...

- O primeiro SGBD comercial surgiu em 1960
- Com o tempo, surgiram padrões para descrever as estruturas de dados: os modelos de dados.
- E surgiu o conceito de metadados: a descrição do banco de dados, segundo um modelo de dados.

# Historicamente...

- Então, o que é um banco de dados?
  - Coleção de dados coerente e logicamente relacionados com algum significado associado
  - Projetado, construído e populado com dados que atendem a um propósito e audiência específicos
  - Representa algum aspecto do mundo real, chamado de minimundo.

# Arquivos versus SGBD's

Processamento tradicional em arquivos	SGBD	Vantagens do SGBD
Definição dos dados associado ao código dos programas da aplicação	Metadados	Eliminação de redundâncias
Dependência entre aplicação e dados	Independência entre aplicações e dados	Eliminação de redundâncias e Facilidade de manutenção
Representação de dados em nível físico	Representação conceitual através de dados e programas	Facilidade de manutenção
Cada visão é implementada por módulos específicos	Permite múltiplas visões	Facilidade de consultas

# Quando usar SGBD

- Quando for essencial:
  - Controle de redundância
  - Controle de consistência e integridade
  - Acesso multiusuário
  - Compartilhamento de dados
  - Controle de acesso e segurança
  - Controle de recuperação e restauração
  - Consultas eficientes

# Arquivos criados durante a aula:

- Informações dos alunos:

Nome	E-mail	Telefone	Empresa de interesse	Representante da empresa
Vanessa Ferreira	<a href="mailto:van@gmail.com">van@gmail.com</a>	99123-1234	Digisystem	Miriam
Jacqueline Prates	<a href="mailto:jac@hotmail.com">jac@hotmail.com</a>	3456-1234		
Giuliana Miniguiti	<a href="mailto:gmin@gmail.com">gmin@gmail.com</a>	9234-2678	Easynvest	Vitor
Guilherme Raulino	<a href="mailto:guir@hotmail.com">guir@hotmail.com</a>	99567-3489	Easynvest	Vitor
Matheus Bolognini	<a href="mailto:mat@gmail.com">mat@gmail.com</a>	99345-6789	Easynvest	Vitor
Vinicius Volpe	<a href="mailto:vinv@hotmail.com">vinv@hotmail.com</a>	99453-2378	Totvs	Rafael

- Informações dos componentes de um grupo:

Nome	E-mail	Telefone	Empresa de interesse	Representante da empresa
Giuliana Miniguiti	<a href="mailto:gmin@gmail.com">gmin@gmail.com</a>	9234-2678	Easynvest	Vitor
Matheus Bolognini	<a href="mailto:mat@gmail.com">mat@gmail.com</a>	99345-6789	Easynvest	Vitor

# Arquivos criados durante a aula:

- No slide anterior, temos a figura de 2 arquivos criados durante a aula:
  - Um com informações dos alunos
  - Outro com as informações dos componentes de um grupo
- A ideia desses arquivos é mostrar o problema que surge quando se armazena as informações sem muito planejamento
- Antes da existência dos bancos de dados, era muito comum o uso de arquivos para armazenar os dados, e na maioria das vezes, sem um planejamento prévio adequado
- No exemplo desses 2 arquivos criados em aula, vemos o problema da redundância dos dados:
  - Se o telefone da Giuliana for alterado, temos que alterar em 2 planilhas.
  - Se o representante da Easynvest for alterado, temos que alterar em várias linhas das 2 planilhas.

# Modelo de dados relacional

- Esse é o modelo de dados mais utilizado
- Nesse modelo, os dados são armazenados em tabelas (ou relações)
- Por exemplo: **tabela ALUNO**:
  - Cada linha representa um aluno diferente

The diagram shows a table with five columns: Nome, E-mail, Telefone, Empresa de interesse, and Representante da empresa. The table has seven rows of data. A green oval labeled 'Dado' points to the value 'Vanessa Ferreira' in the Nome column of the first row. A red oval labeled 'Tupla ou registro (linha)' encloses the entire first row. A blue oval labeled 'Campo ou coluna' points to the 'Representante da empresa' column. The table data is as follows:

Nome	E-mail	Telefone	Empresa de interesse	Representante da empresa
Vanessa Ferreira	<a href="mailto:van@gmail.com">van@gmail.com</a>	99123-1234	Digisystem	Miriam
Jacqueline Prates	<a href="mailto:jac@hotmail.com">jac@hotmail.com</a>	3456-1234		
Giuliana Miniguiti	<a href="mailto:gmin@gmail.com">gmin@gmail.com</a>	9234-2678	Easynvest	Vitor
Guilherme Raulino	<a href="mailto:guir@hotmail.com">guir@hotmail.com</a>	99567-3489	Easynvest	Vitor
Matheus Bolognini	<a href="mailto:mat@gmail.com">mat@gmail.com</a>	99345-6789	Easynvest	Vitor
Vinicio Volpe	<a href="mailto:vinv@hotmail.com">vinv@hotmail.com</a>	99453-2378	Totvs	Rafael

# Banco de dados Relacional x NoSQL

- SQL (Structured Query Language – Linguagem Estruturada para Consulta)
  - Linguagem padrão utilizada para manipular bancos de dados relacionais
  - Os diversos SGBDs utilizam SQL como linguagem padrão para criação dos bancos, inserção, consulta e manutenção dos dados
- NoSQL (Not Only SQL – Não Apenas SQL)
  - Classe de banco de dados não relacionais, muito utilizados atualmente, principalmente para dados “Big Data”
- Veja classificação de sistemas de bancos de dados mais utilizados em:
  - <https://db-engines.com/en/ranking>

# Modelo de dados relacional

- Com relação aos arquivos criados em aula, como tirar a redundância?
- O correto seria criarmos uma nova tabela, apenas com os dados da Empresa (nome, representante).
- E a tabela Aluno teria uma “referência” ao código da empresa que está nessa nova tabela.
- Veja no próximo slide...

# Modelo de dados relacional

- Informações da tabela Aluno:

Nome	E-mail	Telefone	Empresa de interesse
Vanessa Ferreira	<a href="mailto:van@gmail.com">van@gmail.com</a>	99123-1234	1
Jacqueline Prates	<a href="mailto:jac@hotmail.com">jac@hotmail.com</a>	3456-1234	
Giuliana Miniguiti	<a href="mailto:gmin@gmail.com">gmin@gmail.com</a>	9234-2678	2
Guilherme Raulino	<a href="mailto:guir@hotmail.com">guir@hotmail.com</a>	99567-3489	2
Matheus Bolognini	<a href="mailto:mat@gmail.com">mat@gmail.com</a>	99345-6789	2
Vinicius Volpe	<a href="mailto:vinv@hotmail.com">vinv@hotmail.com</a>	99453-2378	3

- Informações da tabela Empresa:

Código da empresa	Nome da empresa	Representante
1	Digisystem	Miriam
2	Easynvest	Vitor
3	Totvs	Rafael

- Dessa forma, elimina-se a redundância dos dados
- Se o representante da Easynvest for alterado, preciso alterar apenas em um lugar (na tabela Empresa)

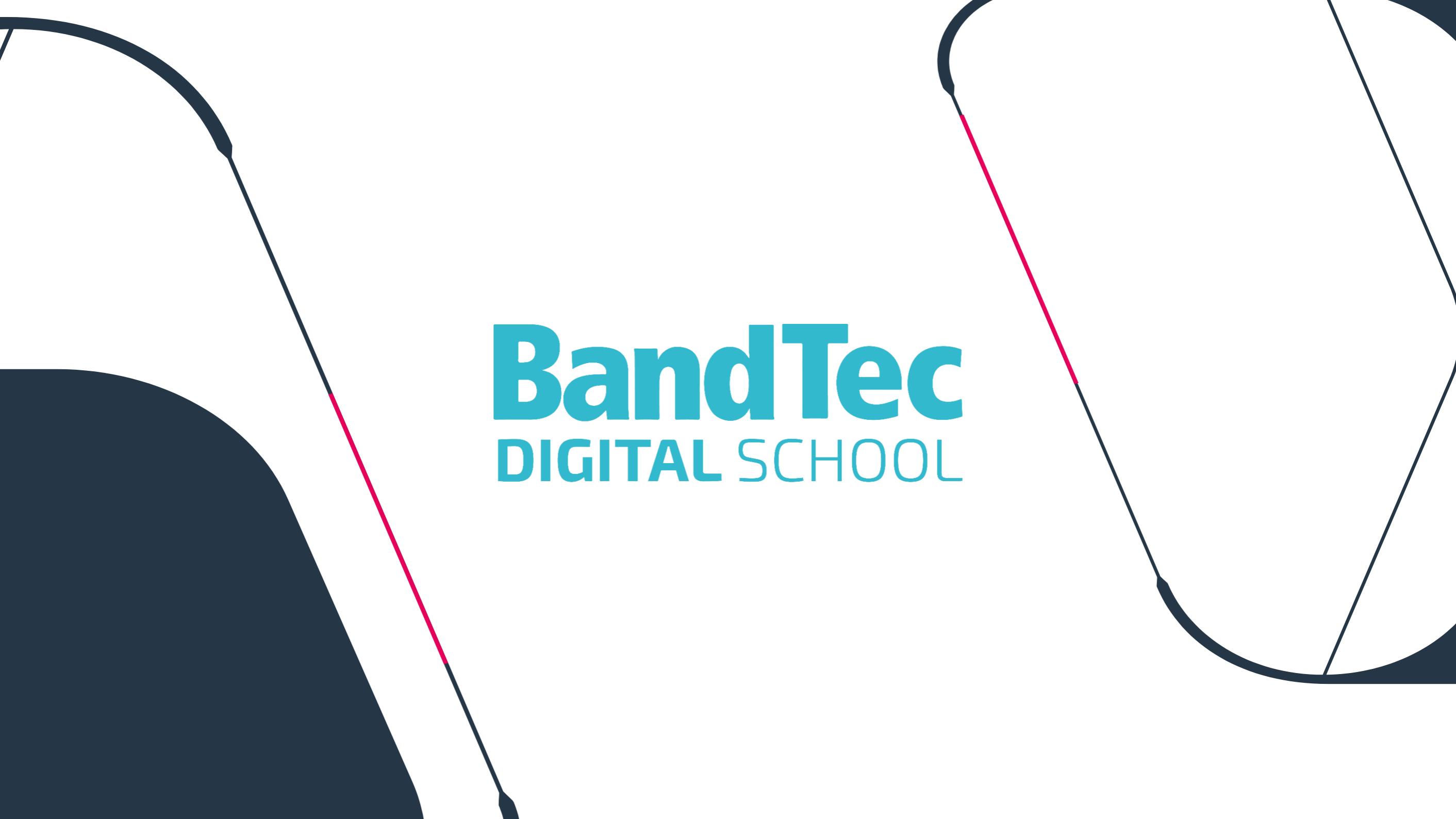
## Exercício

- Criar uma tabela (planilha) com os dados dos alunos: nome, RA, telefone, e-mail, instituição de origem (“link” para a planilha 3), empresa de interesse (“link” para a planilha 2), hobby.
- Criar uma tabela (planilha) com os dados das empresas de interesse dos alunos: nome, representante.
- Criar uma tabela (planilha) com os dados da instituição de origem: nome, bairro.

Obrigada!

**BandTec**  
DIGITAL SCHOOL

Em caso de dúvidas, entre em contato com:  
[celia.taniwaki@bandtec.com.br](mailto:celia.taniwaki@bandtec.com.br)



**BandTec**  
**DIGITAL SCHOOL**

## **BD – Banco de Dados**

Aula02 – Conceitos (Continuação)

# Recapitulando a aula passada

- Dado, informação, conhecimento
- Como se guardavam os dados antes dos bancos de dados ?
- Qual era o problema?
- Banco de dados
- SGBD
- Arquivos vs SGBD

# Modelagem de Dados

- A modelagem de dados é uma técnica utilizada para:
  - Conhecer melhor o contexto de negócio.
  - Retratar os dados que suportam esse contexto de negócio.
  - Projetar o banco de dados.
  - Promover o compartilhamento dos dados e a integração dos sistemas por meio da reutilização de estruturas de dados comuns.
  - Contribuir para que a perspectiva da organização a respeito dos seus dados seja unificada.

# Verificação da pesquisa

- SGBDs : hierárquico, em rede, relacional, orientado a objeto.

# Tipos de SGBD

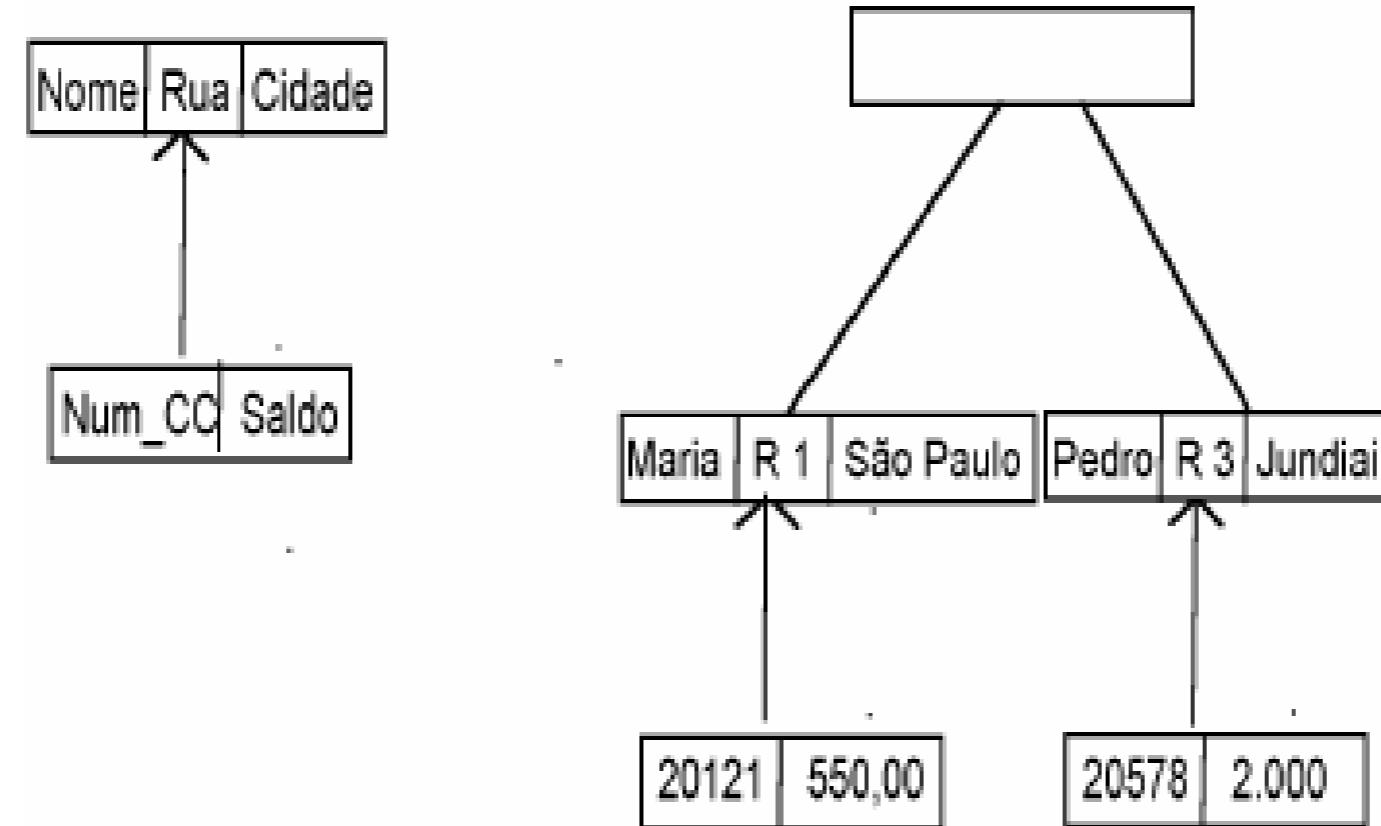
- Modelo hierárquico
- Modelo em rede
- Modelo relacional
- Modelo orientado a objeto

# Modelo Hierárquico

- Primeiro a ser reconhecido como um modelo de dados.
- Representação hierárquica das informações.
- Dados são estruturados em árvores ou hierarquias.
- Cada nó da árvore corresponde à ocorrência de registros (coleção de campos).
- Registro-pai e registros-filhos.
- Ligação – associação entre 2 registros.
- Sistema comercial: IMS (Information Management System) da IBM.

# Modelo Hierárquico

- Exemplo de estrutura do modelo hierárquico
- Conta corrente – endereço

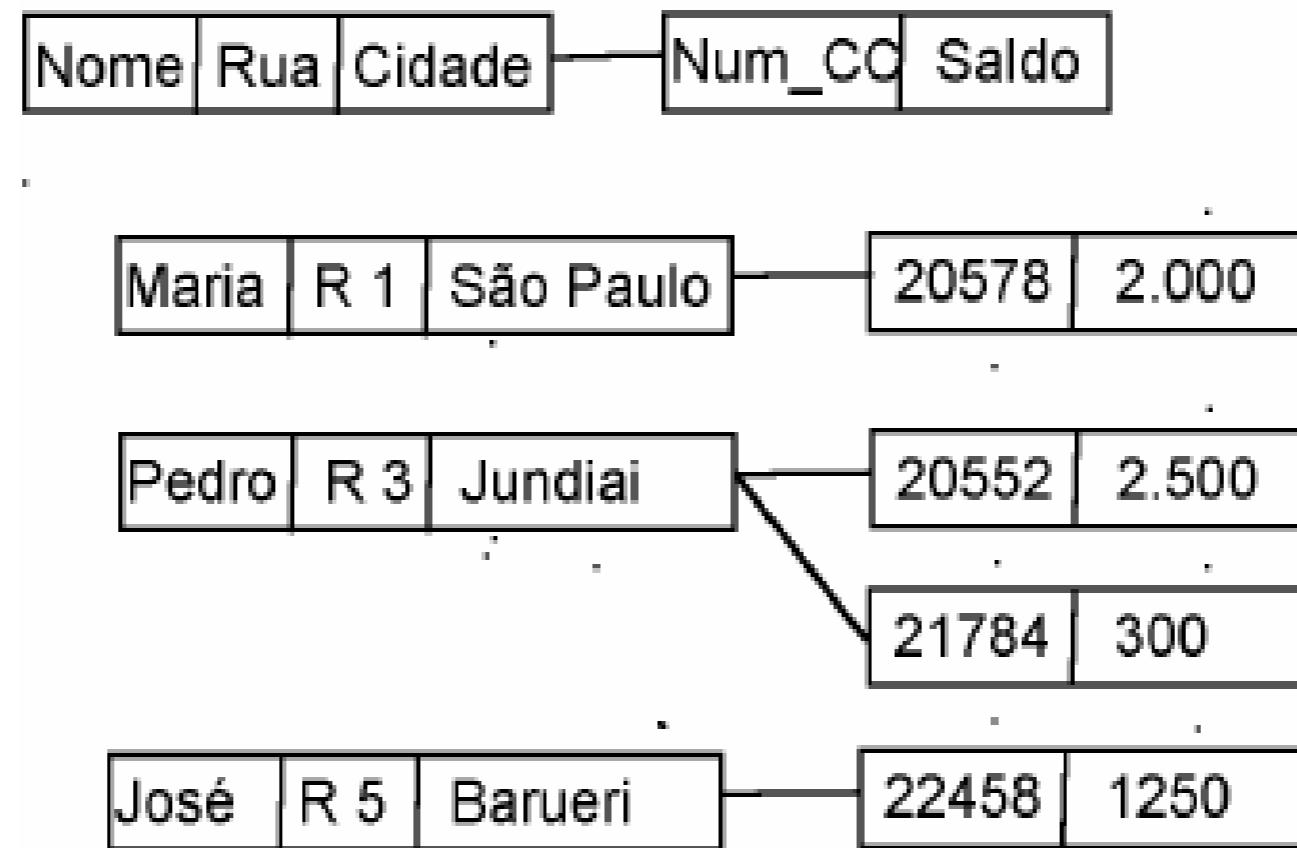


# Modelo em Rede

- Extensão ao modelo hierárquico.
- Eliminou a hierarquia
- Um registro pode estar envolvido em várias associações
- Representado graficamente por grafos.
- Padronizado pela CODASYL (Conference on Data Systems Languages)

# Modelo em Rede

- Exemplo de estrutura do modelo em rede
- Conta corrente – endereço



# Modelo Relacional

- Surgiu para
  - Aumentar a independência dos dados
  - Prover um conjunto de funções para armazenamento e recuperação de dados
- Criado por Edgar Codd, em 1970, tendo como base a teoria dos conjuntos e a álgebra relacional
- Flexível e adequado para solucionar vários problemas na concepção e implementação da base de dados
- Estrutura fundamental: relação (tabela)
- Relação é constituída por um ou mais atributos (campos)

# Modelo Relacional

- Exemplo de tabelas do modelo relacional
- Conta corrente / cliente

Cod_Cliente	Nome	Rua	Cidade
1	Pedro	A	São Paulo
2	Maria	B	Jundiaí

Num_CC	Saldo
20121	1200
21582	1320
21352	652

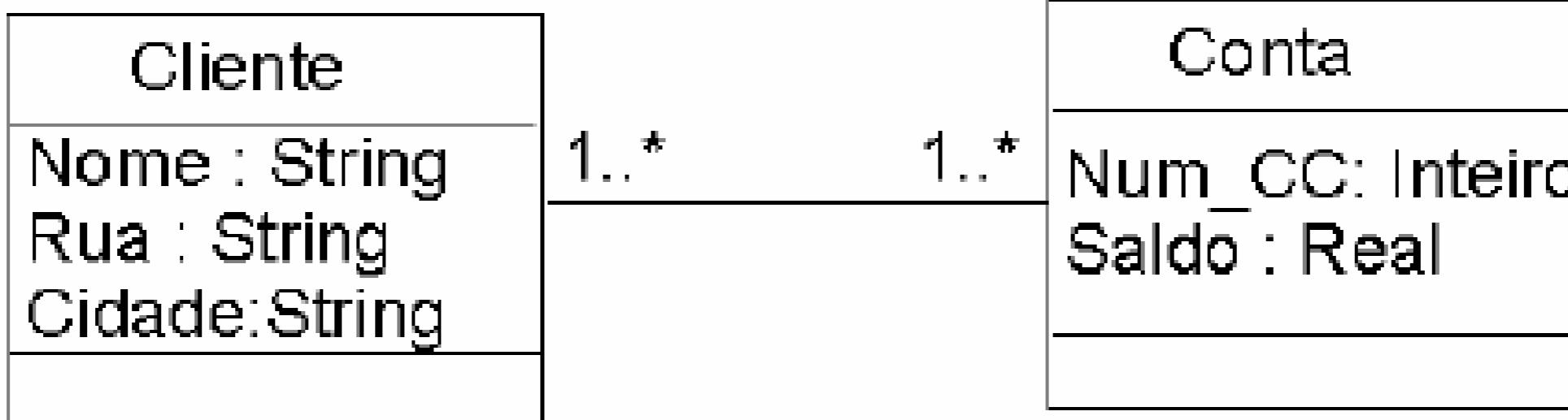
Cod_Cliente	Num_CC
1	20121
2	21582
2	21352

# Modelo Orientado a Objetos

- Comercialmente viável em meados de 1980.
- Surgimento motivado em função dos limites de armazenamento e representação semântica impostas no modelo relacional
  - Ex: sistemas de informações geográficas (tipos complexos de dados)
- Uso de linguagens de programação orientadas a objetos.
- Atualmente, usados em aplicações especializadas
- Representados por diagramas de classes UML (Unified Modeling Language)

# Modelo Orientado a Objetos

- Exemplo de diagrama de classes UML
- Conta corrente – endereço



# Quando não usar um SGBD?

- Aplicações de banco de dados simples e bem definidas, que provavelmente não sofrerão muitas mudanças
- Requisitos rigorosos, de tempo real (podem não ser atendidos pelo SGBD)
- Sistemas embarcados com capacidade de armazenamento limitada
- Nenhum acesso de múltiplos usuários aos dados

# Quais são os atores nessa área?

- Administrador de banco de dados (DBA)
  - Autoriza o acesso ao banco de dados
  - Gerencia e monitora seu uso
  - Adquire recursos de software e hardware
- Projetistas de banco de dados
  - Identifica os dados a serem armazenados
  - Escolhe estruturas apropriadas para representar e armazenar esses dados
- Usuários finais
  - Acessam o banco de dados

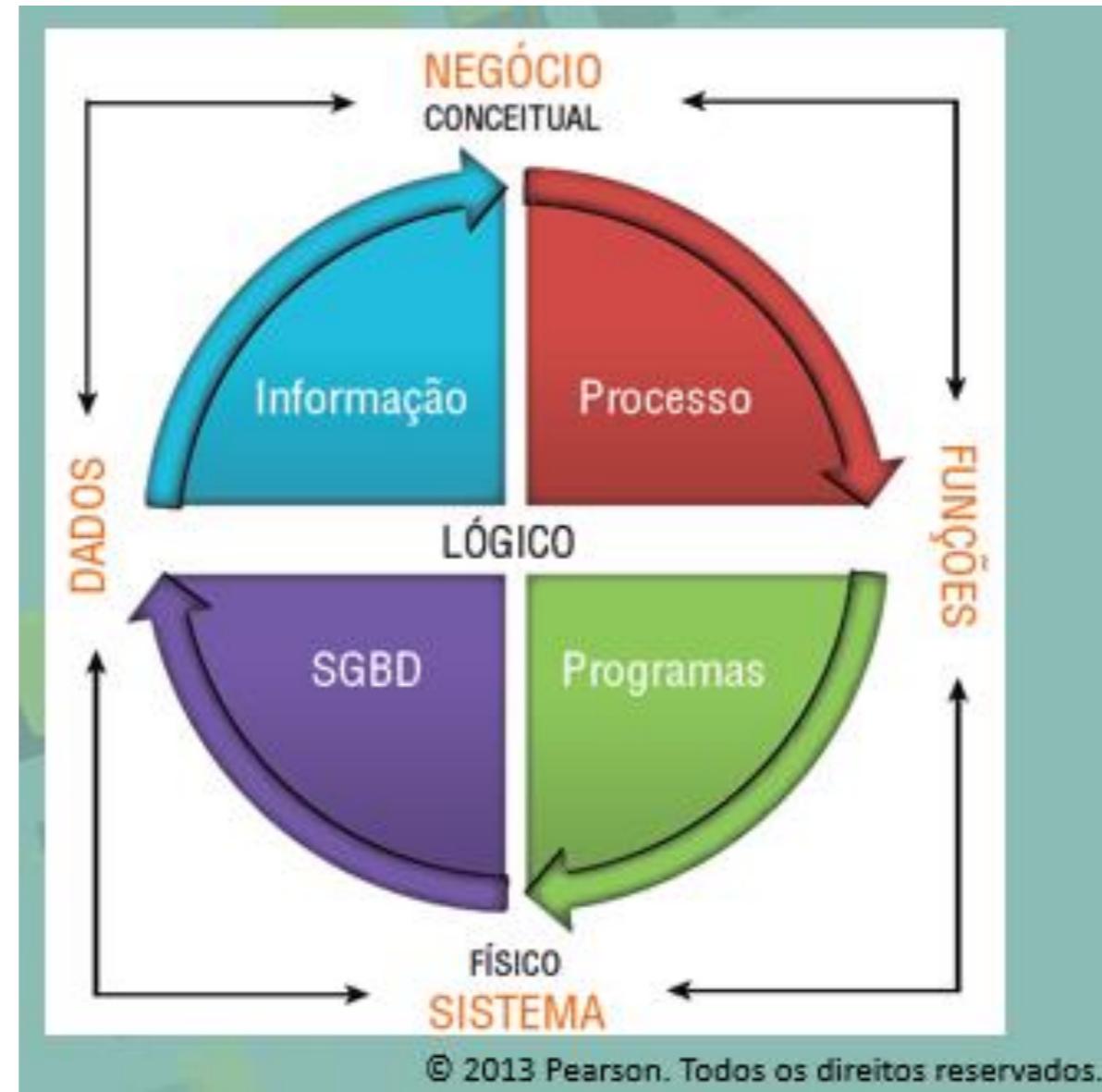
# Quais são os atores nessa área?

- Analistas de sistemas
  - Identificam as necessidades dos usuários finais
  - Modelam e especificam os sistemas
- Programadores de aplicações
  - Implementam essas especificações
  - Testam
  - Documentam
  - Realizam manutenção

# Quem trabalha nos bastidores?

- Projetistas e implementadores de sistemas de SGBD
  - Projetam e implementam os módulos e as interfaces do SGBD
- Desenvolvedores de ferramentas
  - Projetam e implantam ferramentas, como por exemplo: monitoramento de desempenho, simulação e geração da dados para testes
- Operadores e pessoal de manutenção
  - Responsáveis pela execução e manutenção do ambiente de hardware e software para o sistema de banco de dados

# Visão macro do projeto de banco de dados



# Modelos de dados

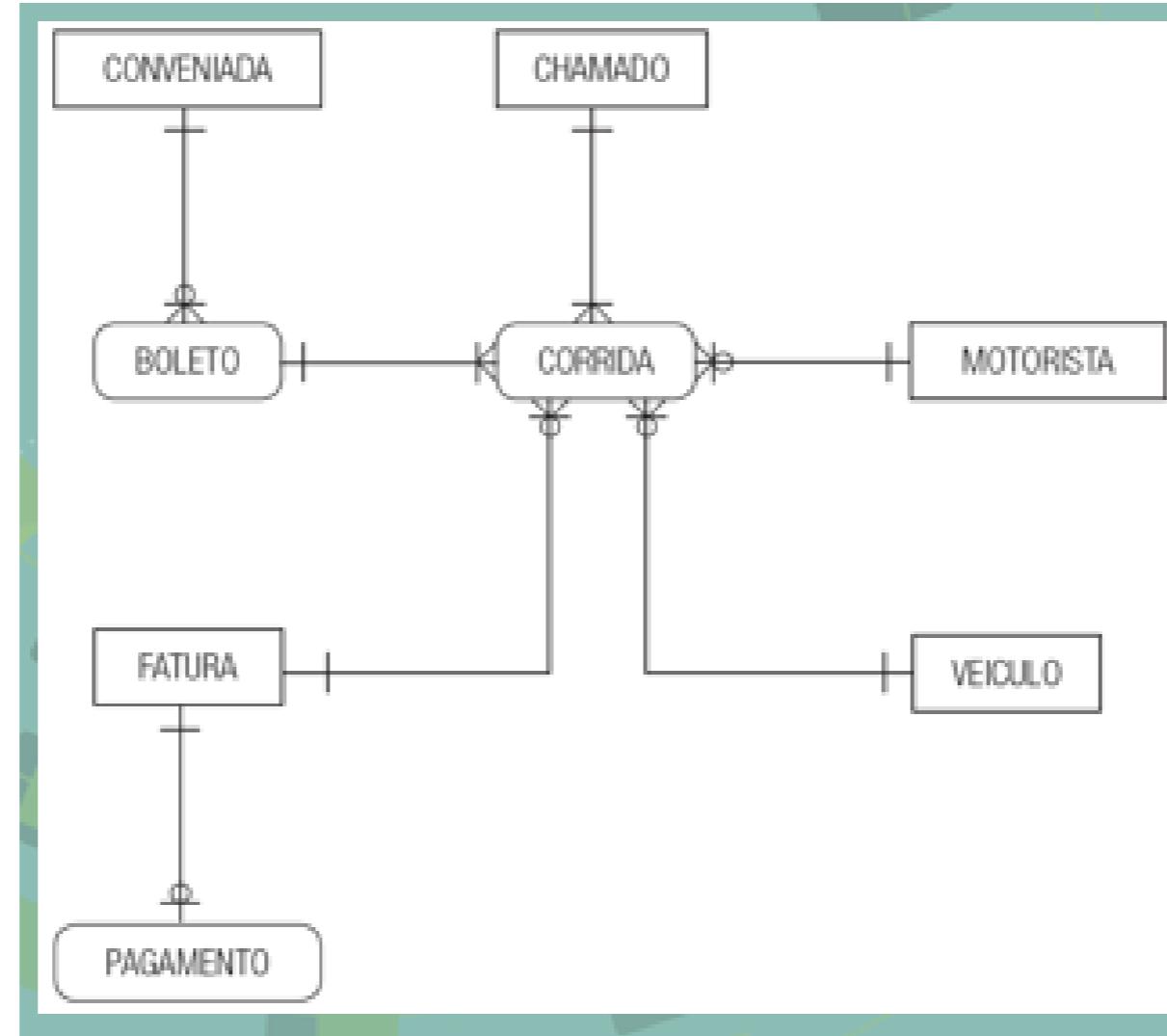
- Modelo conceitual
- Modelo lógico
- Modelo físico

# Modelo conceitual

- Visão de alto nível do banco de dados
- Representa as informações que existem no contexto do negócio
- Funções:
  - Entender o funcionamento de processos e regras do negócio
  - Expressar as necessidades de informações da empresa
  - Facilitar a comunicação entre usuários e área de TI
  - Definir abrangência do sistema, escopo do sistema, estimar custos e prazos de elaboração do projeto
  - Avaliar soluções de software

# Modelo conceitual

- Exemplo – Cenário Rádio Taxi On-line



# Modelo conceitual

- Visão:

**Negócio:** empresa de táxi do segmento de prestação de serviços de táxi, para o transporte de pessoas, encomendas e malotes.

Modelo conceitual	
Informação	Processo
Ficha de emprego: nome, data de nascimento, endereço residencial, telefone residencial, telefone celular, carteira de trabalho, CPF e carteira de habilitação.	Admissão de motorista: para admissão, o candidato deve ter experiência comprovada de, no mínimo, 2 anos como taxista, possuir carteira de habilitação da categoria e não apresentar pontos na carteira de habilitação.

# Modelo lógico

- Representa a versão do modelo conceitual que pode ser apresentada ao SGBD
- Reflete as propriedades necessárias para a tradução do modelo conceitual de forma que possa ser descrito e interpretado por SGBD

# Modelo lógico

**Negócio:** empresa de táxi do segmento de prestação de serviços de táxi, para o transporte de pessoas, encomendas e malotes.

<b>Modelo lógico</b>		
	<b>Estrutura de dados</b>	<b>Programas</b>
<b>DADOS</b>	<p>Estrutura de dados do motorista<sup>1</sup>:</p> <p>1 .    Numero_Matricula_Motorista 2 .    Nome_Motorista 3 .    Data_Nascimento 4 .                 Sexo 5 .                 Numero_CPF 6 . . . . .</p>	<p>Estrutura de uma aplicação, em que devem ser ilustradas as regras para validação dos atributos.</p> <p>1. Obter data da primeira comprovação de trabalho na profissão de motorista.</p> <p>2. Calcular tempo de experiência, subtraindo a data da primeira comprovação de trabalho, pela da data de hoje.</p> <p>3. Caso o tempo de experiência for menor que dois anos, exibir a mensagem: "Tempo de experiência inferior ao mínimo exigido".</p>

# Modelo físico

- Representa a estrutura para armazenamento físico dos dados, em termos computacionais

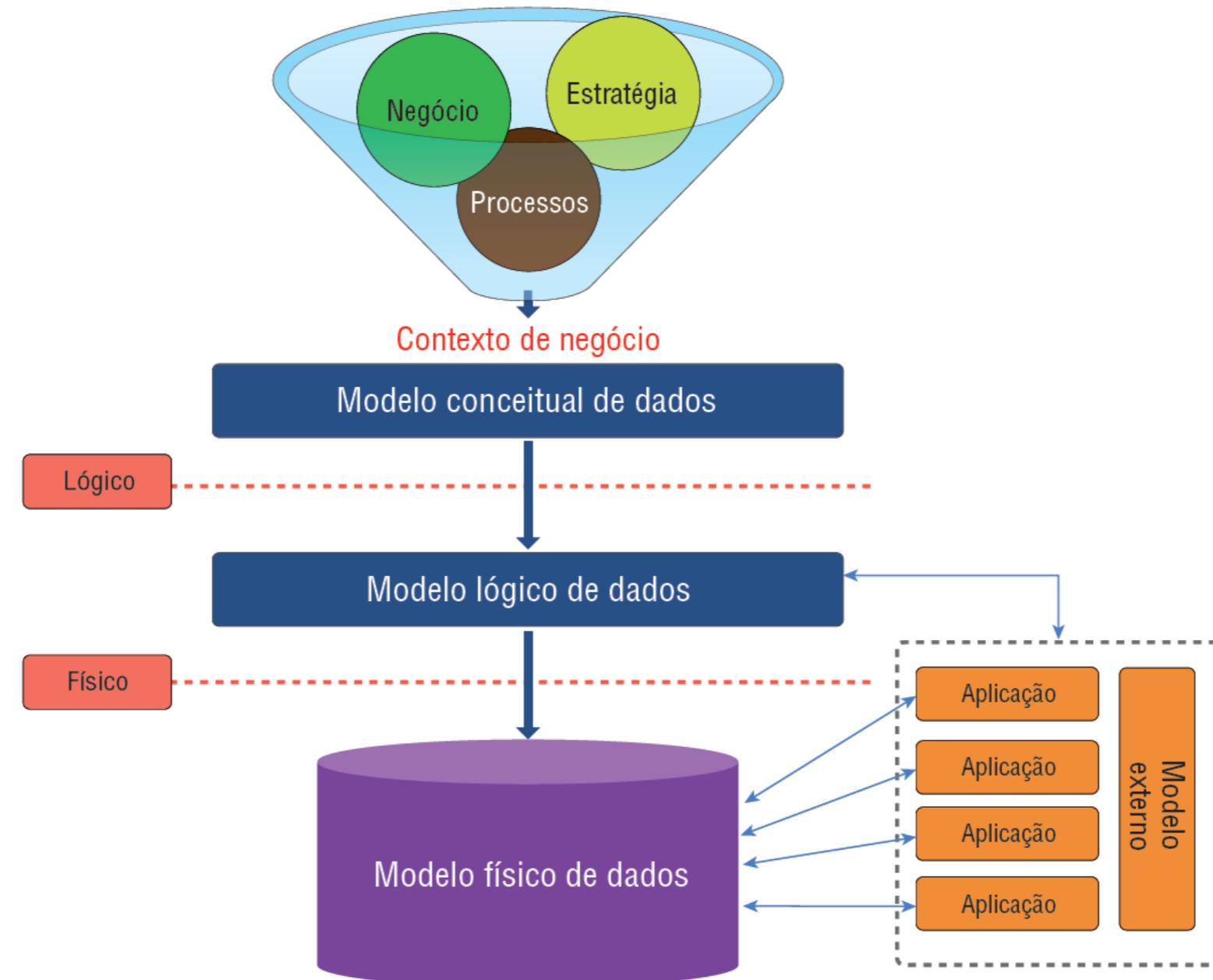
# Modelo físico

**Negócio:** empresa de táxi do segmento de prestação de serviços de táxi, para o transporte de pessoas, encomendas e malotes.

Modelo fisico	
Tabela	MOTORISTA (
num_motorista	NUMBER(5) NOT NULL,
nom_motorista	VARCHAR2(50) NOT NULL,
dat_nascimento	DATE NOT NULL,
idt_sexo	VARCHAR2(1) - ADD CONSTRAINT CK_PES_FIS_IDT_SEXO CHECK (Idt_sexo IN ('F', 'M')),
num_CPF	NUMBER(11) NOT NULL
.....	)

**Sistemas:** o objetivo do *Sistema de Cobrança Rádio Táxi On-line* é automatizar registro, controle e acompanhamento de chamados, bem como armazenar as informações da emissão de faturas e cobrança das conveniadas.

# Modelagem de Dados



# Pesquisar os termos a seguir relacionados a SQL

- chave primária (primary key)
- chave estrangeira (foreign key)
- valor NULL
- DDL (relacionado à linguagem SQL)
- DML

# Verificação da pesquisa (continuação)

- chave primária (primary key)
  - Campo que identifica de forma única uma tupla ou registro da tabela. Por exemplo: campo RA da tabela Aluno.
- chave estrangeira (foreign key)
  - Campo que se refere à chave primária de uma outra tabela. Por exemplo: campo Empresa de interesse da tabela Aluno, que contém o código da empresa (chave primária da tabela Empresa). Veja slide da Aula01.
- valor NULL
  - Quando um determinado campo, para um registro (linha) não tem nenhum valor, ele recebe o valor NULL.

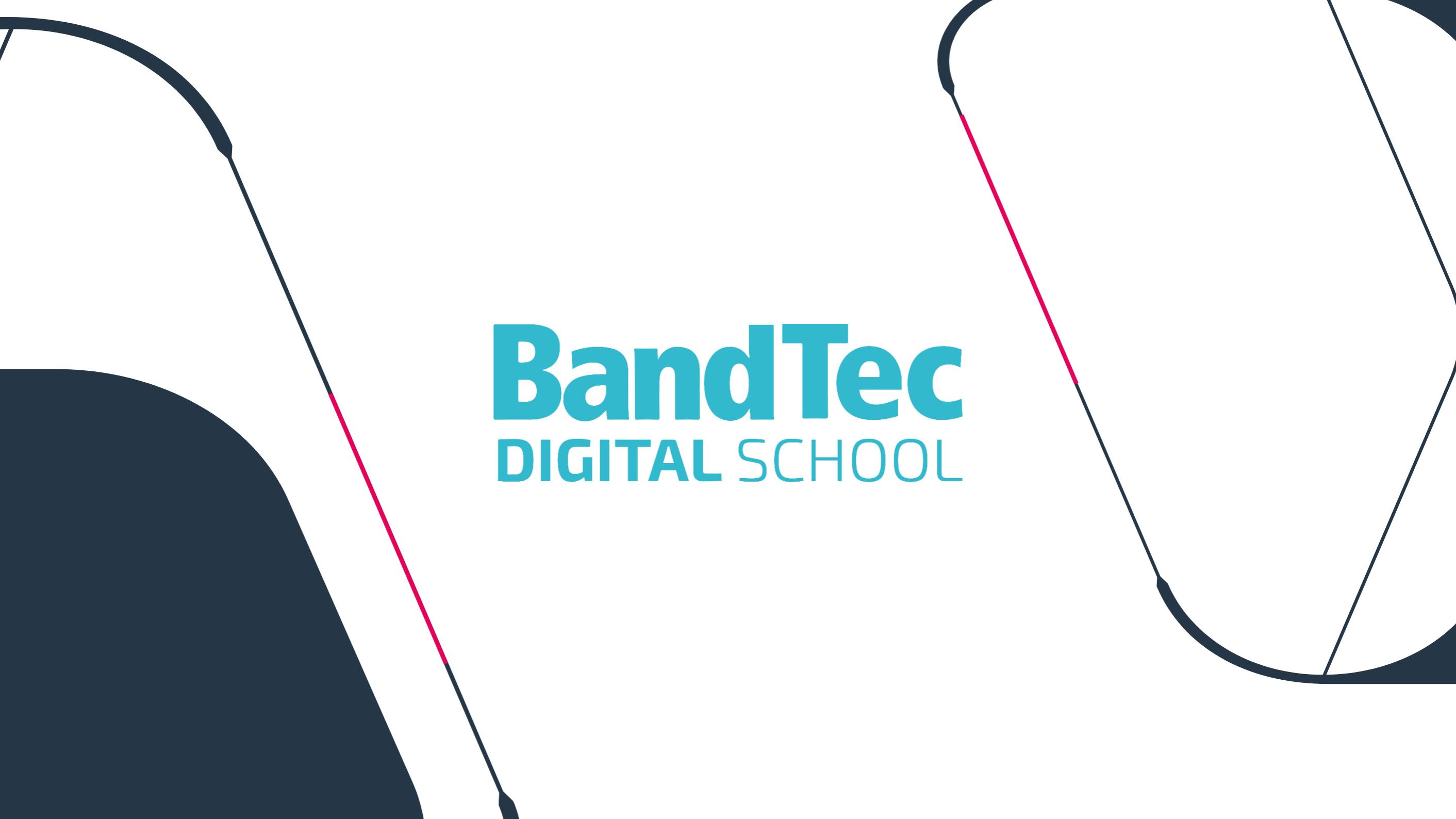
# Verificação da pesquisa (continuação)

- DDL (relacionado à linguagem SQL)
  - Data Definition Language – grupo de instruções do SQL para criar tabelas, alterar a estrutura das tabelas ou eliminar tabelas.
    - Instruções CREATE, ALTER, DROP
- DML (relacionado à linguagem SQL)
  - Data Manipulation Language – grupo de instruções do SQL para manipular as tabelas, ou seja, para inserir dados, atualizar os dados, excluir dados, consultar dados
    - Instruções INSERT, UPDATE, DELETE, SELECT

Obrigada!

**BandTec**  
DIGITAL SCHOOL

Em caso de dúvidas, entre em contato com:  
[celia.taniwaki@bandtec.com.br](mailto:celia.taniwaki@bandtec.com.br)



**BandTec**  
**DIGITAL SCHOOL**



## **BD – Banco de Dados**

Aula03 – Comandos SQL (MySQL Server)

# SQL – Structured Query Language

- Desenvolvido no início dos anos 1970, pelo Departamento de pesquisas da IBM
  - Interface para o sistema de banco de dados relacional System R
  - Inicialmente chamava-se SEQUEL (Structured English QUERy Language)
  - A partir de 1977, passou a ser chamada de SQL
  - Query = consulta, em inglês

# SQL – Structured Query Language

- Em 1986, o Instituto Nacional Americano de Padrões (ANSI) juntamente com a ISO (International Standards Organization) publicaram o padrão de linguagem SQL-86 ou SQL-1
  - Linguagem padrão adotada para Bancos de Dados Relacionais
  - Os vários SGBDs relacionais passaram a utilizar SQL
- A SQL-86 passou por revisões:
  - SQL-92 ou SQL-2, em 1992
  - SQL-99 ou SQL-3, em 1999
  - SQL-2003, em 2003

# SQL – Categorias de instruções

- DDL (relacionado à linguagem SQL)
  - Data Definition Language – grupo de instruções do SQL para criar tabelas, alterar a estrutura das tabelas ou eliminar tabelas.
    - Instruções CREATE, ALTER, DROP
- DML (relacionado à linguagem SQL)
  - Data Manipulation Language – grupo de instruções do SQL para criar manipular as tabelas, ou seja, para inserir dados, atualizar os dados, excluir dados, consultar dados
    - Instruções INSERT, UPDATE, DELETE, SELECT

# Comandos SQL – MySQL e SQL Server

- No MySQL Server, todos os comandos devem ser finalizados por ponto e vírgula (;)
- No SQL Server, da Microsoft, não é obrigatório o ponto e vírgula no final do comando

# Criando um banco de dados (Schema)

- Comando:

**CREATE DATABASE nome-do-banco;**

- No MySQL Workbench, os bancos de dados aparecem na parte esquerda da tela, na janela Schemas
- As tabelas são criadas dentro de um banco de dados. Para isso, devemos selecionar o banco de dados que queremos utilizar

# Selecionando um banco de dados (Schema)

- Comando:  
**USE nome-do-banco;**
- No MySQL Workbench, é possível selecionar o banco apenas dando um duplo clique com o mouse em cima do nome do banco na janela SCHEMAS

# Criando uma tabela

- Comando:

```
CREATE TABLE nome-da-tabela (  
    nome-campo1 tipo-campo1,  
    nome-campo2 tipo-campo2,  
    .....  
    nome-campoN tipo-campoN  
);
```

- A tabela será criada dentro do banco de dados selecionado, com os campos definidos no comando.
- No comando acima, é possível acrescentar restrições aos campos, como PRIMARY KEY, etc.

# Criando uma tabela (exemplo)

- Exemplo feito em aula:

```
CREATE TABLE Aluno (  
    ra INT PRIMARY KEY,  
    nome VARCHAR(40),  
    bairro VARCHAR(40)  
);
```

- O comando acima criará a tabela Aluno, com 3 campos: ra, nome e bairro.
- O campo ra terá um valor numérico e inteiro e esse campo será a chave primária da tabela.
- Os valores dos campos nome e bairro serão caracteres.

# Campos com valores caracteres

- Campo com valores caracteres como o nome e o bairro podem ser definidos com o tipo CHAR ou VARCHAR
- Diferença entre CHAR e VARCHAR:
  - Quando se define que o nome é CHAR(10), então todos os campos desse tipo terão 10 caracteres, mesmo que o nome inserido tenha menos do que 10 caracteres
    - Ex: 'Bruno' será armazenado como 'Bruno 'O campo nome terá 5 espaços em branco para completar 10 caracteres
  - Quando se define que o nome é VARCHAR(10), então todos os campos terão no máximo 10 caracteres. Se o nome inserido tiver menos do que 10 caracteres, o campo conterá apenas os caracteres inseridos
    - Ex: 'Bruno' será armazenado como 'Bruno'

# Visualizando ou listando os dados da tabela

- Comando:

```
SELECT * FROM nome-da-tabela;
```



\* significa “**todas as colunas**”

- O comando acima exibe todos os dados de uma tabela
- Quando a tabela acabou de ser criada e ainda não tem dados, o comando exibirá apenas os títulos das colunas

# Inserindo dados na tabela

- Comando:

**INSERT INTO** nome-da-tabela

**VALUES** (valor-campo1, valor-campo2, ..., valor-campoN);

- O comando acima vai inserir os dados dentro dos parênteses na tabela, preenchendo um novo registro (nova “linha”) na tabela
- A ordem que os dados devem aparecer dentro do parênteses deve corresponder à ordem da criação dos campos no comando CREATE TABLE

# Inserindo dados na tabela (exemplo)

- Exemplo:

```
INSERT INTO Aluno  
VALUES (51000, 'Maria' , 'Paraíso');
```

- O comando acima vai inserir os dados correspondentes a um aluno, de ra= 51000, nome = 'Maria', bairro = 'Paraíso'
- Repare que para o valor int, não é preciso aspas
- E que para o valor do tipo varchar, é preciso aspas
  - Tanto aspas simples quanto aspas duplas são aceitas, mas **acostume-se a utilizar aspas simples**

# Inserindo dados de mais de uma linha na tabela

- Exemplo:

```
INSERT INTO Aluno
```

```
VALUES (51001, 'José' , 'Tatuapé'), (51002, 'Claudio', 'Cambuci'),  
(51003, 'Ana', 'Saúde'), (51004, 'Marina', 'Jabaquara') ;
```

- Pode-se inserir de uma só vez os dados de vários alunos, como no exemplo acima (inserção de 4 alunos)
- É preciso tomar cuidado para colocar entre parênteses os dados de cada aluno, na ordem em que foi criada a tabela Aluno, separados por vírgula.
- Cada conjunto de parênteses deve ser separado por vírgula.

# Visualizando os dados apenas de algumas colunas

- Exemplo 1:

**SELECT** nome **FROM** Aluno;

O comando acima exibe apenas a coluna nome da tabela Aluno.

- Exemplo 2:

**SELECT** nome, bairro **FROM** Aluno;

O comando acima exibe apenas as colunas nome e bairro da tabela Aluno.

- Exemplo 3:

**SELECT** bairro, ra **FROM** Aluno;

O comando acima exibe apenas as colunas bairro e ra da tabela Aluno, nessa ordem.

# Visualizando os dados apenas de algumas linhas

- Comando:

**SELECT \* FROM nome-da-tabela WHERE condição;**



**pode ser \***  
**ou o(s) nome(s)**  
**da(s) coluna(s)**  
**desejada(s)**



**utiliza-se o WHERE**  
**para apresentar a condição**  
**para “filtrar” as linhas desejadas**

- As linhas que satisfazem a condição (colocada após o WHERE) são exibidas pelo comando.

# Visualizando os dados apenas de algumas linhas

- Exemplo 1:

```
SELECT * FROM Aluno WHERE ra = 51002;
```

O comando acima exibe os dados do aluno de RA 51002

- Exemplo 2:

```
SELECT * FROM Aluno WHERE ra >= 51002;
```

O comando acima exibe os dados dos alunos de RA  $\geq$  51002

- Exemplo 3:

```
SELECT * FROM Aluno WHERE ra <> 51002;
```

O comando acima exibe os dados dos alunos de RA diferente de 51002

Obs.: o MySQL e o SQL Server aceitam também != como sinal de “diferente”, mas o padrão é <>

# Visualizando os dados apenas de algumas linhas

- Exemplo 4:

```
SELECT * FROM Aluno WHERE ra BETWEEN 51002 AND 51004;
```

O comando acima exibe os dados dos alunos de RA 51002 (inclusive) a 51004 (inclusive)

- Exemplo 5:

```
SELECT * FROM Aluno WHERE ra >= 51002 AND ra <= 51004;
```

O comando acima é equivalente ao do Exemplo 4.

# Visualizando os dados apenas de algumas linhas

- Exemplo 6:

```
SELECT * FROM Aluno WHERE nome LIKE 'M%';
```

O comando acima exibe os dados dos alunos cujo nome começa com M.

Quando se especifica um padrão como 'M%', utiliza-se o LIKE.

O sinal de % representa zero ou mais caracteres.

Dessa forma, esse comando procurará os nomes que tenham a primeira letra M e depois pode vir uma quantidade qualquer de caracteres, e não importa quais caracteres.

# Visualizando os dados apenas de algumas linhas

- Exemplo 7:

```
SELECT * FROM Aluno WHERE nome LIKE '%a';
```

O comando acima exibe os dados dos alunos cujo nome começa termina com a.

O sinal de % representa zero ou mais caracteres.

Dessa forma, esse comando procurará os nomes que tenham uma quantidade qualquer de caracteres, e não importa quais caracteres, desde que no final tenha a letra a

# Visualizando os dados apenas de algumas linhas

- Exemplo 8:

```
SELECT * FROM Aluno WHERE nome LIKE '_n%';
```

O comando acima exibe os dados dos alunos cujo nome tenha a letra n como segunda letra.

O sinal de \_ representa apenas um caractere.

Dessa forma, esse comando procurará os nomes que tenham um caractere qualquer, seguido da letra n e depois do n pode vir uma quantidade qualquer de caracteres, e não importa quais caracteres

# Visualizando os dados ordenados por outra coluna

- Quando executamos o SELECT, os dados são exibidos de forma ordenada pela coluna que é a chave primária da tabela.
- No exemplo da tabela Aluno, os dados são exibidos ordenados pelo RA, que é a chave primária.

**SELECT \* FROM** Aluno **ORDER BY** nome-da-coluna;

OU

**SELECT \* FROM** Aluno **ORDER BY** nome-da-coluna **ASC**;

O comando acima exibe os dados dos alunos ordenados pela coluna especificada, em ordem ascendente (do menor para o maior, ou em ordem alfabética)

- Se quiser que a ordem seja descendente:

**SELECT \* FROM** Aluno **ORDER BY** nome-da-coluna **DESC**;

# Visualizando os dados ordenados por outra coluna

- Exemplo 9:

```
SELECT * FROM Aluno ORDER BY nome;
```

O comando acima exibe os dados dos alunos ordenados pelo nome (em ordem ascendente – ou seja, em ordem alfabética).

- Exemplo 10:

```
SELECT * FROM Aluno ORDER BY bairro DESC;
```

O comando acima exibe os dados dos alunos ordenados pelo bairro, em ordem descendente – ou seja, do 'Z' ao 'A'.

# Exibindo a descrição da tabela

- Comando:

**DESC** nome-da-tabela;

OU

**DESCRIBE** nome-da-tabela;

- Esse comando exibe uma descrição da tabela

- Exemplo:

**DESC** Aluno;

- A execução desse comando produz esse resultado:

The screenshot shows a MySQL Workbench interface with a 'Result Grid' tab selected. The grid displays the structure of the 'Aluno' table. The columns are labeled: Field, Type, Null, Key, Default, and Extra. The data rows are:

	Field	Type	Null	Key	Default	Extra
▶	ra	int(11)	NO	PRI	NULL	
	nome	varchar(50)	YES		NULL	
	bairro	varchar(40)	YES		NULL	

# Excluindo a tabela

- Comando:  
**DROP TABLE** nome-da-tabela;
- Exemplo:  
**DROP TABLE** Aluno;
- A execução desse comando excluirá a tabela Aluno do banco de dados.

# Excluindo o banco de dados

- Comando:

**DROP DATABASE nome-do-banco;**

- Exemplo:

**DROP DATABASE BancoAluno;**

- A execução desse comando excluirá o banco de dados (Schema) BancoAluno.

# Alterando o valor de algum dado já inserido

- Comando:

**UPDATE** nome-da-tabela **SET** coluna-a-ser-alterada = novo-valor  
**WHERE** condição;

- As linhas que satisfazem a condição (colocada após o WHERE) terão o valor alterado na coluna-a-ser-alterada, especificada no comando.
- Exemplo:

**UPDATE** Aluno **SET** bairro = 'Tatuapé' **WHERE** ra = 51003;

O comando acima altera o valor do bairro do aluno de RA 51003 para 'Tatuapé'

# Alterando o valor de algum dado já inserido

- Pode-se alterar o valor de mais de uma coluna numa mesma linha em um só comando.
- Exemplo:

```
UPDATE Aluno SET nome = 'Ana Maria', bairro = 'Tatuapé'  
WHERE ra = 51003;
```

O comando acima altera o valor do nome e do bairro do aluno de RA 51003.

- **ATENÇÃO: Se não for colocado a cláusula WHERE, todas as linhas da tabela serão afetadas por este comando.**

(o MySQL tem uma proteção contra isso, mas ela pode ser desabilitada, e outros SGBDs não têm essa proteção)

# Excluindo uma ou mais linhas da tabela

- Comando:  
**DELETE FROM** nome-da-tabela **WHERE** condição;
- As linhas que satisfazem a condição (colocada após o WHERE) serão excluídas da tabela
- Exemplo: **DELETE FROM** Aluno **WHERE** ra = 51003;  
O comando acima exclui da tabela a linha referente ao aluno de RA 51003.
- **ATENÇÃO: Se não for colocado a cláusula WHERE, todas as linhas da tabela serão afetadas por este comando.**  
(o MySQL tem uma proteção contra isso, mas ela pode ser desabilitada, e outros SGBDs não têm essa proteção)

# MySQL – Comandos UPDATE e DELETE

- O MySQL tem uma proteção que **não permite** que o **UPDATE** e o **DELETE FROM** sejam executados **sem que se coloque a cláusula WHERE**.
- O MySQL também obriga que se utilize na condição do **WHERE** a **coluna que é a chave primária da tabela**. Isso vale para os comandos **UPDATE** e **DELETE FROM**
- Isso já não ocorre no SQL Server (que será usado no Azure)
- No MySQL, essa proteção pode ser desabilitada, caso o usuário queira.
- Por isso, é bom tomar cuidado em **não esquecer** de colocar **WHERE** nos comandos **UPDATE** e **DELETE FROM**.

# Alterando o valor de algum dado já inserido

- Pode-se alterar o valor de mais de uma coluna numa mesma linha em um só comando.
- Exemplo:

```
UPDATE Aluno SET nome = 'Ana Maria', bairro = 'Tatuapé'  
WHERE ra = 51003;
```

O comando acima altera o valor do nome e do bairro do aluno de RA 51003.

- **ATENÇÃO:**

**Se não for colocado a cláusula WHERE, todas as linhas da tabela serão afetadas por este comando.**

(No MySQL, há uma proteção que não deixa isso acontecer)

Obrigada!

**BandTec**  
DIGITAL SCHOOL

Em caso de dúvidas, entre em contato com:  
[celia.taniwaki@bandtec.com.br](mailto:celia.taniwaki@bandtec.com.br)