CAPÍTULO I – Bancos de Dados

Dados e Informações

"**Dados** são conjuntos de fatos distintos e objetivos, relativos a eventos" [1].
O funcionamento das organizações gera grandes quantidades de dados. Mas, para o processo de tomada de decisões, os dados não apresentam uma importância direta, pois as pessoas podem não compreender o significado desses dados, da maneira como eles são apresentados.
Quando compreendemos o significado dos dados, esses se transformam em **Informações**, a matéria prima para a tomada de decisão.
Daí podemos deduzir algumas coisas importantes. Para a tomada de decisão dentro das organizações, necessitamos obter informações, mas essas dependem fundamentalmente dos dados existentes interna e externamente a essas organizações. Além disso, os dados são a matéria prima e o resultado das transações que as organizações realizam no seu dia-a-dia, para realizar os seus negócios.
Portanto, é vital para que as organizações possam funcionar normalmente que dados, que apresentem importância para essas organizações, sejam armazenados de forma organizada e segura.

Banco de Dados

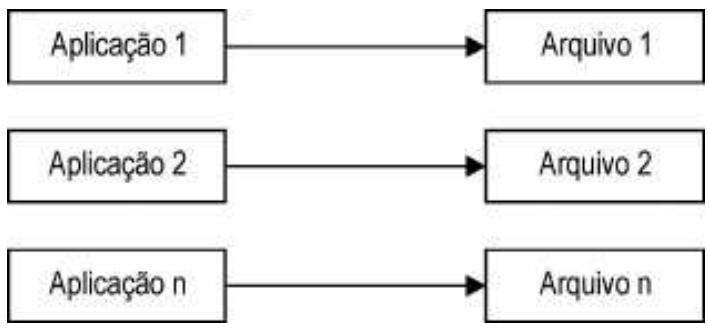
Banco de Dados é um "aplicativo que armazena dados de uma forma organizada e que permita a recuperação desses dados".
Os Bancos de Dados podem ser **Integrados** e **Compartilhados**. Eles são Integrados, quando representam a unificação de diferentes arquivos de dados e são Compartilhados quando permitem que mais do que uma aplicação acesse os dados armazenados nele.
Além disso, eles devem procurar evitar a **Redundância** e a **Inconsistência**. A Redundância, ou seja, o armazenamento do mesmo dado em diferentes locais, é algo útil para efeito de segurança (funciona como um backup dos dados), mas ela pode apresentar um problema inaceitável para o processo de armazenamento de dados: a Inconsistência.
Um tipo de inconsistência que os Bancos de Dados apresentam é quando os mesmos tipos de dados são armazenados em locais diferentes (por exemplo, o endereço de um cliente), mas apresenta valores diferentes em cada um dos locais (por exemplo, o cliente atualizou o seu endereço, mas o processo só ocorreu em um dos locais de armazenamento dos dados).
Dessa forma, podemos deduzir que, nos Bancos de Dados, a redundância pode ser aceita (apesar de apresentar outro problema, que é o aumento do espaço necessário para armazenar esses dados), mas a inconsistência deve ser evitada de qualquer maneira.

---

[1] DAVENPORT, Thomas H.; PRUSAK, Laurence. *Conhecimento Empresarial: Como as organizações gerenciam o seu capital intelectual*. Rio de Janeiro: Campus, 1998. p. 2.

Histórico dos Bancos de Dados

Inicialmente, os dados necessários ao funcionamento das organizações eram armazenados em **Arquivos**, que são agrupamentos de dados armazenados em algum dispositivo de armazenamento físico, e para acessar esses dados e utilizá-los no dia-a-dia, eram criadas **Aplicações** (programas de computador).



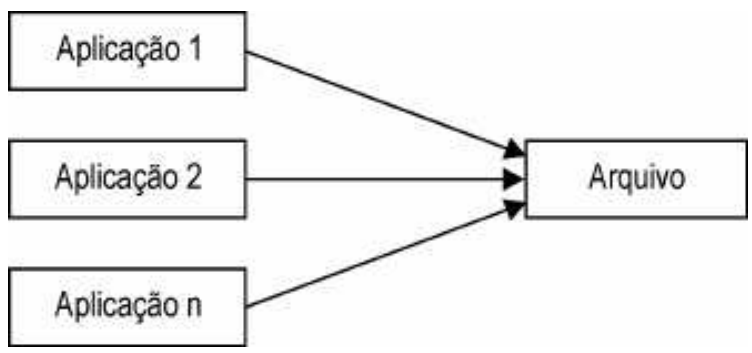
O problema dessa abordagem é que cada tipo de aplicação possuía o seu arquivo de dados. Vejamos, imagine uma empresa de manufatura, onde existem arquivos armazenando dados referentes ao projeto do produto (Departamento de Engenharia), aos detalhes dos componentes do produto (Departamento de Compras) e aos detalhes de montagem do produto (Departamento de Produção).

Cada um desses departamentos possui a sua aplicação específica e, portanto, seu arquivo de dados específico. Mas, quando um novo produto é criado pelo Departamento de Engenharia é necessário que os dados dos componentes desse produto sejam enviados para os Departamentos de Compra e de Produção.

Como esse modelo de armazenamento não permite que uma aplicação acesso o arquivo de dados da outra, os dados eram transferidos na forma de papel impresso para o outro departamento e digitados novamente, na outra aplicação.

Essa redundância de dados não existia devido a preocupação com a segurança, mas sim por uma questão técnica, o que levava a grandes quantidades de dados idênticos sendo armazenados em diversos arquivos diferentes. Além desse problema, pois nesta época os dispositivos de armazenamento eram muito caros, a possibilidade de ocorrerem inconsistência nos dados era enorme.

A solução apresentada para esse problema foi unificar arquivos que armazenavam os mesmos dados e permitir um acesso compartilhado a esse arquivo, por parte das aplicações.



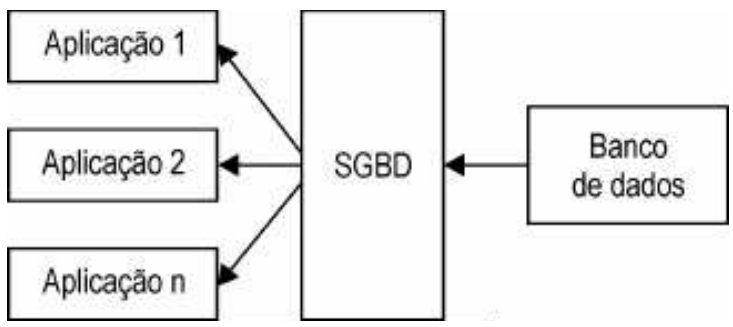
Isso resolveu o problema da redundância e da inconsistência, mas apresentou outros. O primeiro deles é que permitindo o acesso de todos ao mesmo arquivo, todas as aplicações eram obrigadas a tratar dados que podiam não apresentar importância para ela e, até mesmo, nem deveriam ser acessados. Observe o exemplo a seguir.

Imagine um arquivo armazenando os dados de um empregado. Nesse arquivo estão armazenados todos os dados necessários sobre as pessoas que trabalham na organização. Agora imagine que existem dois aplicativos acessando esse arquivo, o aplicativo do Departamento de Recursos Humanos e o aplicativo do Departamento de Operações, que cria uma escala de trabalho para os empregados.

Como todos os aplicativos têm acesso a todos os dados armazenados no arquivo, o Departamento de Operações terá acesso a dados que não tem nenhuma importância para ele, como por exemplo, o número de um determinado documento do empregado, como também terá acesso a dados que são sigilosos e que nem deveriam ser acessados, como por exemplo, o salário do empregado.

O segundo problema é que, qualquer alteração na estrutura do arquivo (estrutura do armazenamento dos dados), que seja feita para adequar o arquivo a uma determinada aplicação, implica na modificação de todas as outras aplicações para se adequarem a essa mudança também.

Para resolver isso, surgiu o conceito do **Sistema Gerenciador de Bancos de Dados (SGBD)**[2], que consiste em uma aplicação que administra o acesso aos dados dos arquivos de dados. O SGBD funciona como um "intermediário" entre as aplicações e os arquivos de dados (o Banco de Dados), mantendo a unificação dos arquivos e o compartilhamento do acesso, além de evitar a inconsistência dos dados, mas controlando o acesso a esses dados.



Esse controle permite que a aplicação acesse somente os dados que sejam importantes para o seu funcionamento (mesmo que existam mais dados armazenados) e inibe o acesso àqueles dados que deveriam ser sigilosos para aquela aplicação.

Além disso, ele resolve outro problema existente nos dois modelos anteriores. Antes do surgimento dos SGBD, as aplicações eram responsáveis por determinar o armazenamento físico dos dados, ou seja, se preocupar em como armazenar os dados nos dispositivos de armazenamento. Essa tarefa foi passada para os SGBDs, o que facilita em muito a vida dos programadores de aplicações.

Modelos de Bancos de Dados

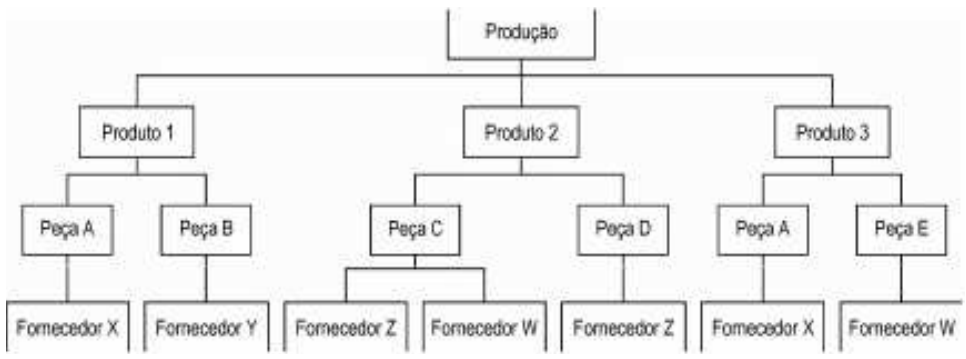
Como vimos, com o surgimento dos SGBDs, para a criação de uma aplicação que acesse Banco de Dados não é mais necessário se preocupar com o armazenamento físico desses dados, dessa forma era necessário que esses dados fossem modelados (mostrados) de uma forma compreensível para o desenvolvedor.

[2] Em inglês **DBMS (Database Management System)**

Existem diversas formas de mostrar esses dados, esses são os **Modelos de Bancos de Dados**. Lembrando que um modelo é apenas a forma como os dados são mostrados e não como eles são fisicamente armazenados (preocupação do SGBD e não do desenvolvedor).

Modelo Hierárquico

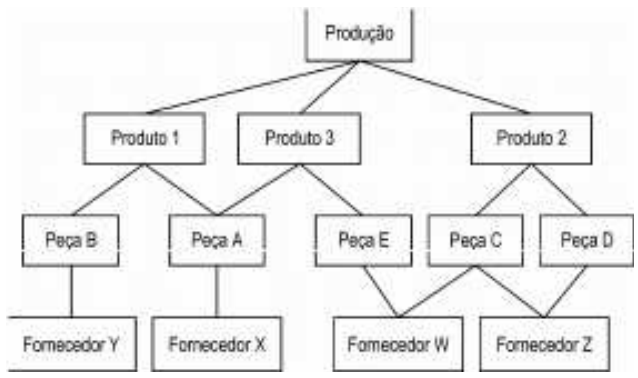
Um dos primeiros modelos a surgir foi o **Modelo Hierárquico**. Esse modelo mostrava os dados na forma de uma "árvore" invertida, conforme figura abaixo.



Nele os dados eram ligados na forma de dados pai (mais acima na estrutura) e dados filhos (mais abaixo na estrutura). Isso gerava uma estrutura muito grande e com muitos dados sendo apresentados de forma repetida, o que poderia gerar dificuldade de compreensão. Observe na figura acima o Fornecedor W, ele aparece duas vezes, porque ele é fornecedor das peças C e E.

Modelo em Rede

Para resolver o problema de representação apresentado pelo Modelo Hierárquico, surgiu o **Modelo de Rede**. Esse modelo é muito semelhante ao anterior, mas com a diferença que cada dado é apresentado somente uma vez.



Por exemplo, observe novamente o Fornecedor W, ele agora aparece somente uma vez, mas com duas ligações (uma para cada peça que ele fornece). O problema desse modelo é que ele pode se tornar bastante confuso, quando as linhas que ligam os diferentes níveis começam a se cruzar.

Esses dois modelos apresentavam os dados de uma forma mais próxima ao armazenamento físico dos arquivos. Com o surgimento dos SGBDs, era necessário mostrar os dados de outra forma.

Modelo Relacional

O **Modelo Relacional** foi criado para permitir que os dados fossem apresentados de uma forma mais próxima da realidade e mais adequada para o uso dos SGBDs. Os dados são apresentados na forma de **Tabelas**, sendo que cada linha da tabela é um relacionamento entre um conjunto de valores.
Alguns conceitos relacionados a esse modelo são apresentados a seguir.

Domínios

Domínios são conjuntos de valores possíveis para uma entidade[3] e seus atributos. Abaixo exemplos de domínios da tabela **Roupa** (para os atributos **Código**, **Nome**, **Local** e **Cor**).

Código
001
002
003
004
005

Nome
Camisa
Calça
Saia
Bermuda
Camiseta

Local
São Paulo
Campinas
Rio de Janeiro
Vitória
Manaus
Peruíbe
Bauru

Cor
Branco
Preto
Vermelho
Verde
Azul
Amarelo
Rosa

Uma representação dos dados usando o Modelo Relacional ficaria da seguinte forma (tabela **Roupa**).

Código	Nome	Local	Cor
001	Camisa	São Paulo	Branco
002	Calça	Peruíbe	Rosa
003	Camisa	São Paulo	Amarelo
004	Bermuda	Vitória	Verde
005	Camiseta	Manaus	Verde

Observe que nem todos os valores dos domínios foram utilizados, apenas aqueles que tem relação com os objetos lógicos representados (lembre que essa tabela é uma representação lógica dos dados referentes aos objetos físicos).
Nesse modelo as tabelas apresentam as seguintes propriedades:

1) As linhas são distintas.
2) Os nomes das colunas são únicos.
3) A ordem das linhas é irrelevante.
4) A ordem das colunas é irrelevante.

Cada coluna de uma tabela do Modelo Relacional, que correspondem aos atributos, é chamada de **Campo** e cada linha de uma tabela é chamada de **Registro** (também conhecida como **Tupla**).[4]

---

[3] Conjunto de objetos da realidade modelada sobre os quais se deseja manter informações no banco de dados.
[4] Na literatura, quando fazemos o projeto lógico do banco de dados utilizamos as denominações **Atributo** e **Tupla**, e quando fazemos o projeto físico utilizamos **Campo** e **Registro**. Na verdade são a mesma coisa.

Chaves

Chave é o campo (atributo) que identifica de maneira unívoca o registro (tupla), ou seja, para que não existem duas linhas com os dados iguais em todas as colunas. Na tabela exemplo, o campo **Código** é usado para essa identificação. Esse campo é conhecido como **Chave Primária (PK – Primary Key)** da tabela **Roupa**.

Existem casos onde é preciso utilizar mais do que um campo como Chave. Esses campos unidos que representam a chave primária são conhecidos como **Chave Primária Composta** (observe o exemplo abaixo da tabela **Produto**).

Nota Fiscal	Código	Quantidade
N1	P1	100
N1	P2	200
N1	P3	300
N2	P4	400
N2	P1	100
N3	P2	200
N3	P3	300

Obs: como o campo Chave Primária identifica o registro, ele não pode apresentar valor nulo (nulo não significa zero, mas sim sem nenhum valor armazenado).

Chave Estrangeira

Observe as tabelas **Roupa** e **Fornecedor** abaixo:

Roupa				
PK			FK	
Código	Nome	Cor	Cód_Forn	Local
R1	Camisa	Preto	F2	São Paulo
R2	Saia	Preto	F1	Bauru
R3	Calça	Branco	F1	São Paulo
R4	Camisa	Verde	F3	São Paulo
R5	Bermuda	Branco	F2	Campinas

Fornecedor		
PK		
Código	Nome	Cidade
F1	ABC	São Paulo
F2	XYZ	Salvador
F3	WWW	São Paulo

No campo **Cód_Forn** da tabela **Roupa** são encontrados valores correspondentes aos valores encontrados no campo **Código** da tabela **Fornecedor**, ou seja, existe um relacionamento entre esses dois campos. Só é possível acrescentar valores em **Cód_Forn** que tenham correspondência em **Código** (tabela **Fornecedor**).

O campo **Cód_Forn** é conhecido como **Chave Estrangeira (FK - Foreign Key)**, ou seja, é um campo que não é chave em uma tabela (tabela **Roupa**) que se relaciona com um campo chave em outra tabela (tabela **Fornecedor**).

As chaves primárias e as chaves estrangeiras fornecem os meios para representar os **Relacionamentos** entre tabelas no Modelo Relacional.

Visão

Visão é uma tabela que é derivada de outras tabelas (uma ou mais) e não existe por si só. As visões são "instantâneos" dos dados armazenados nas tabelas, portanto não existem fisicamente.
As visões são obtidas por meio de **Consultas** e estas são realizadas por meio de **Linguagens de Consulta**, sendo a mais famosa o **SQL (Structured Query Language)**.
No exemplo abaixo é apresentada a tabela **Pessoa_Brasil** que é uma visão da tabela **Pessoa**, na qual são selecionados os registros onde o campo **País** é igual ao valor "*Brasil*".

Pessoa			
Cod	**Nome**	**Cidade**	**País**
01	João	São Paulo	Brasil
02	Peter	Nova Iorque	Estados Unidos
03	Brigitte	Paris	França
04	Maria	Porto Alegre	Brasil
05	Hans	Berlin	Alemanha
06	Joanna	Los Angeles	Estados Unidos

Pessoa_Brasil			
Cod	**Nome**	**Cidade**	**País**
01	João	São Paulo	**Brasil**
04	Maria	Porto Alegre	**Brasil**

Bancos de Dados Multidimensionais

Como já vimos anteriormente, os Bancos de Dados são muito importantes para o processo de tomada de decisões nas organizações, pois eles armazenam os dados das transações que ocorrem no dia-a-dia delas.
Os sistemas tradicionais armazenam os dados do que aconteceu na organização, mas determinados dados vão sendo alterados ao longo do processo (como por exemplo, os dados sobre a quantidade de determinado produto em estoque de um supermercado ou os dados sobre a cotação de uma determinada moeda estrangeira). Esses são os **Sistemas Transacionais** e utilizam um processamento conhecido como **OLTP (On-Line Transaction Processing)**.
Com o tempo, surgiu a necessidade de dados históricos e consolidados (de diferentes fontes) sobre as transações realizadas nas organizações. Isso ocorreu devido à necessidade de dados consolidados para a tomada de decisão em um nível mais estratégico da organização. Surgiu um outro tipo de processamento o **OLAP (On-Line Analytical Processing)**. A tabela abaixo apresenta uma comparação de algumas características dos dois tipos de processamento.

Características	Sistemas Transacionais(OLTP)	Sistemas Analíticos(OLAP)
Exemplos	CRM, ERP, Supply Chain	SIG, SAD, SIE
Atualizações	Mais freqüentes	Menos freqüentes
Tipo de Informação	Detalhes (maior granularidade)	Agrupamento (menor granularidade)
Quantidade de Dados	Poucos	Muitos
Precisão	Dados atuais	Dados históricos
Complexidade do Resultado da Pesquisa (para o negócio)	Baixa	Alta
Terminologia	Linhas e Colunas	Dimensões, Medidas e Fatos

Para apoiar esse tipo de processamento surgiram os **Bancos de Dados Multidimensionais**, também conhecidos como **Data Warehouses**. Um Data Warehouse é um conjunto de bancos de dados, geralmente do modelo relacional, que consolida as informações empresariais, provenientes das mais distintas fontes de dados.

A sua principal característica é ser a consolidação de dados de diferentes bancos de dados transacionais e sempre ter que incorporar um componente temporal. Simplificando para facilitar a compreensão, podemos imaginar diversas tabelas do modelo relacional, agrupadas e sobrepostas, sendo que cada uma dessas tabelas está relacionada com um período de tempo (uma tabela por dia, semana, mês ou ano). É possível imaginar como se as tabelas fossem páginas que vão sendo colocadas umas sobre as outras, representando o instantâneo dos dados consolidados naquele determinado período de tempo.

O processo de construção desse tipo de banco de dados é moroso e complexo, principalmente devido à grande quantidade de dados, provenientes de diversas fontes heterogêneas de dados, algumas vezes inconsistentes e envolvendo várias áreas diferentes da organização.

A dificuldade de consolidação dos dados transacionais é decorrente de:

- Várias fontes heterogêneas de dados;
- Dados de entrada precisam se "limpos", ou seja, adequados a um formato padrão;
- A periodicidade de obtenção dos dados (granularidade) deve ser ajustada;
- Pode ser preciso resumir os dados; e
- Necessidade de um componente temporal, nem sempre presente nas fontes de dados.

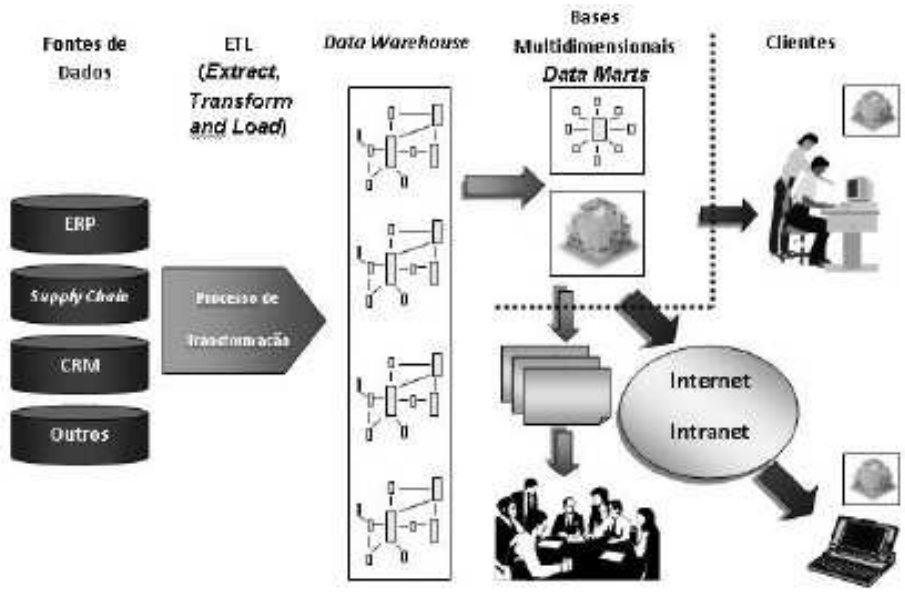
Como exemplo, podemos mostrar algumas situações comuns encontradas nas fontes de dados para a montagem de bancos de dados multidimensionais. São elas:

- Mesmos dados com nomes de campos diferentes;
- Dados diferentes com o mesmo nome de campo;
- Dados exclusivos de uma determinada aplicação; e
- Chaves diferentes para um mesmo tipo de tabela.

Tão importante quanto saber quais dados armazenar, é saber quando e quais dados remover do Data Warehouse, porque com o tempo esse tipo de banco de dados começa a ficar muito grande. Algumas estratégias existentes, e que podem ser utilizadas em conjunto, são:

- Resumir dados mais antigos; e
- Armazenar dos dados antigos em mídias de armazenamento mais baratas (como fitas, por exemplo);

Um esquema de banco de dados multidimensional pode ser visto na figura abaixo:



Obs: **Data Marts** podem ser compreendidos como sendo Data Warehouses departamentais ou intermediários.