

Ao final deste assunto, você será capaz de:

- Compreender a arquitetura de esquemas de um sistema de banco de dados;
- Conhecer as linguagens de manipulação de banco de dados;
- Conhecer a evolução histórica dos sistemas de banco de dados, sabendo como diferenciar uma estrutura da outra através de suas particularidades.

1.2.1. Abstração de Dados

O objetivo da arquitetura de esquemas para sistemas de banco de dados é separar o banco de dados das aplicações do usuário por meio de três níveis de esquemas. São eles:

- Esquema interno: Descreve a estrutura física de armazenamento do banco de dados, a sua organização de arquivos e os seus métodos de acesso;
- Esquema conceitual: Descreve a estrutura do banco de dados completo sob o ponto de vista do usuário, escondendo os detalhes de armazenamento e concentrando-se na descrição dos objetos que descrevem os dados (entidades, atributos, relacionamentos, operações do usuário e restrições sobre os dados);
- Esquemas externos: Descrevem as partes do banco de dados que são do interesse de um grupo de usuários, escondendo as demais. Essas são as visões de usuário.

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

Na figura a seguir é apresentado o esquema de três níveis:

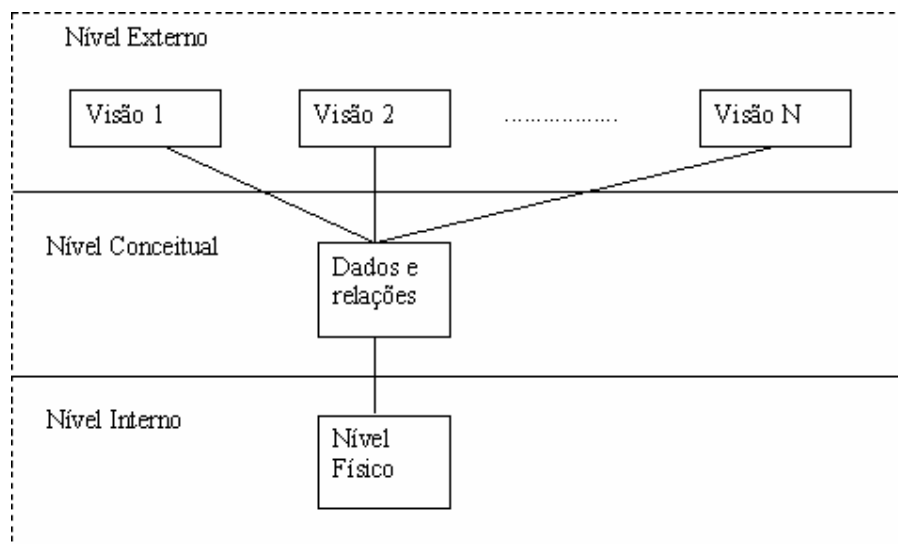


Fig. 1.2. 1 - Arquitetura de Três Níveis ⁽¹⁾.

A arquitetura de três níveis também é conhecida como Arquitetura ANSI/SPARC.

1.2.2. Linguagens de Manipulação de Banco de Dados

A partir do momento em que o projeto do banco de dados está pronto, este precisa ser implementado fisicamente no banco de dados (SGBD) e, a partir daí, os dados já podem ser inseridos no BD. Esses dois processos são efetuados nos BD através do uso de linguagens de manipulação. São elas:

- Linguagem de Definição de Dados (Data Definition Language - DDL):
“Linguagem que define as aplicações, arquivos e campos que irão compor o banco de dados (comandos de criação e atualização da estrutura dos campos dos arquivos)”.
- É a linguagem de definição de dados, através da qual o próprio banco é definido, bem como as suas tabelas e colunas das tabelas.

¹ Korth, H.F. e Silberschatz, A.; Sistemas de Bancos de Dados, Makron Books, 2a. edição revisada, 1994.

É a linguagem que permite especificar o esquema do banco de dados, através de um conjunto de definições de dados.

Onde ficam armazenadas estas definições?

A compilação dos comandos em DDL é armazenada no dicionário de dados.

- *Linguagem de Manipulação de Dados (Data Manipulation Language - DML)*
“Linguagem que permite aos usuários acessar ou manipular dados organizados por um modelo de dados apropriado”.

Linguagem que define os comandos de manipulação e operação dos dados (comandos de consulta e atualização dos dados dos arquivos).

É a linguagem de manipulação dos dados, através da qual o usuário pode criar, obter ou eliminar os dados do seu banco de dados. Esta linguagem é de uso geral, ou seja, normalmente está disponível para os usuários do BD.

É a linguagem que permite ao usuário acessar ou manipular os dados, vendo-os da forma como são definidos no nível de abstração mais alto do modelo de dados utilizado.

O que é a manipulação de dados?

- A recuperação da informação armazenada no banco de dados;
- A inserção de novas informações no banco de dados;
- A remoção de informações do banco de dados.

Existem dois tipos básicos e DML:

- *Procedimental ou Procedural*: *requer do usuário a especificação de quais dados são desejados e como chegar até eles (Ex: Álgebra Relacional);*
- *Não-Procedimental ou Não-Procedural*: *requer do usuário a especificação de quais dados são desejados, sem especificar como*

chegar até eles (Ex: Cálculo Relacional).

Todo banco de dados tem as suas DDLs e DMLs, sendo que muitas vezes elas ficam escondidas dentro das facilidades de uso do SGBD. Alguns SGBDs mais sofisticados disponibilizam o uso dessas linguagens para os seus usuários.

Com a evolução e facilidade de uso dos seus SGBDs, os fabricantes passaram a construir esses softwares com características e facilidades muito parecidas, porém para acessar os diversos SGBDs o usuário tinha que usar DMLs diferentes.

Com o objetivo de facilitar o uso dos BD relacionais e facilitar a migração e convivência entre os diversos SGBDs buscou-se a padronização da DML. A IBM saiu na frente e em 1986 conseguiu padronizar junto a ANSI (American National Standard Institute - Instituto Nacional Americano de Padrões) o SQL (Structured Query Language) como DML de acesso a SGBD relacional. Ou seja, nesse ano o SQL passou a ser a DML padrão no território americano. Em 1987, a ISO(International Standards Organization - Organização Internacional de Padrões) padronizou o SQL com DML a nível mundial.

1.2.3. Estruturas Lógicas de Banco de Dados

Os SGBDs são implementados pelos fabricantes de acordo com uma determinada estrutura lógica que o fabricante entende ser a melhor para o desenvolvimento e uso de seu produto.

Os SGBD's vem evoluindo desde que surgiu o conceito de banco de dados, partindo de idéias do meio industrial até que se produzisse conceitos no meio acadêmico, os quais vieram solidificar mais essa área de conhecimentos, que mais tarde viria a ser conhecida com Informática, uma ciência e não mais um modo de fazer as coisas.

Como a tecnologia de banco de dados surgiu como uma evolução do conceito de arquivos, os primeiros SGBD's surgiram com o seu embasamento estrutural baseado em registros, ou seja, os dados teriam que ser armazenados e individualizados em grupamentos inter-relacionados.

A Estrutura Lógica de Banco de Dados é basicamente uma forma utilizada para descrever logicamente um Banco de Dados. Não existe uma única forma de representação deste modelo, porém qualquer forma que permita a correta compreensão das estruturas de dados que fazem parte do Banco de Dados pode ser considerada adequada. Vamos descrever sucintamente as diversas formas de abordar os modelos lógicos de dados:

- Abordagem Orientada a Registros: São modelos que representam esquematicamente as estruturas das tabelas de forma bastante próxima a existente fisicamente. Basicamente, são apresentados os registros de cada tabela (inclusive seus campos) e seus relacionamentos elementares. O Modelo Relacional, o Modelo de Rede e o Hierárquico são exemplos deste tipo de representação. Historicamente, os sistemas gerenciadores de banco de dados podem ser classificados da seguinte forma:
 - Sistemas Não-Relacionais;
 - Sistemas Relacionais.
- Abordagem Orientada a Objetos: São modelos que procuram representar as informações através dos conceitos típicos da Orientação a Objetos, utilizando o conceito de Classes que irão conter os objetos. Citamos os Modelos O2 (França); PCTE (Comunidade Européia); Orion (EUA) e o Modelo de Representação de Objetos - MRO (Brasil - USP/São Carlos) como exemplos típicos desta abordagem.

À medida que começou a ganhar espaço entre a comunidade de desenvolvedores de software, a tecnologia de orientação a objetos passou a ser um novo paradigma na área de programação e o meio acadêmico passou a questionar porque os dados necessários a uma aplicação orientada a objetos não poderiam ser armazenados conforme as definições da aplicação (orientados a objeto). Apesar de ser, de fato, o estado da arte em tecnologia de banco de dados, ainda não conseguiu alcançar plenamente os mercados profissionais, mas já vem aos poucos conseguindo adeptos e fornecedores

cada vez mais interessados em preparar seus produtos para o novo paradigma (a orientação a objetos em Bancos de Dados).

Para maiores detalhes, consulte o documento [Banco de Dados Orientado a Objetos – BDOO \(apend-a\)](#).

Abordagem orientada a registros

Sistemas Não-Relacionais

São sistemas de Banco de Dados que não seguem a abordagem relacional.

São eles:

- Sistemas Hierárquicos;
- Sistema em Rede;
- Sistemas de Lista Invertida.

Características comuns:

- Mais antigos;
- Não houve base teórica para sua criação. Ou seja, foram criados como evolução dos sistemas de arquivos. Não foram pensados abstratamente antes da criação de produtos;
- São sistemas baseados em registros, ou seja, cada dado pretende representar um registro;
- Apenas o modelo de dados é implementado no banco. As regras que garantem a integridade dos dados devem ser implementadas por programação. Não há preocupação com integridade referencial.

Sistemas Hierárquicos

Características

- Estrutura de dados em árvores, ou seja, estrutura de níveis, onde o nível zero é a raiz;
- Os dados são divididos em registros que apontam sucessivamente para registros hierarquicamente inferiores e assim por diante. O banco precisa armazenar esses elos como registros pai-filho;
- Não é possível que um filho tenha mais que um pai;
- Implementa automaticamente regras de integridade referencial quando há dependência hierárquica. Não existe filho sem pai, exceto se for raiz;
- Uma mudança em qualquer nível hierárquico implica na reconstrução do banco. Ou seja, fazer o arquivo voltar a ser seqüencial, reconstruí-lo pela DDL e realimentá-lo pela DML;
- É possível construir-se índices para pesquisa mais rápida;
- DDL e DML próprios.

Exemplo:

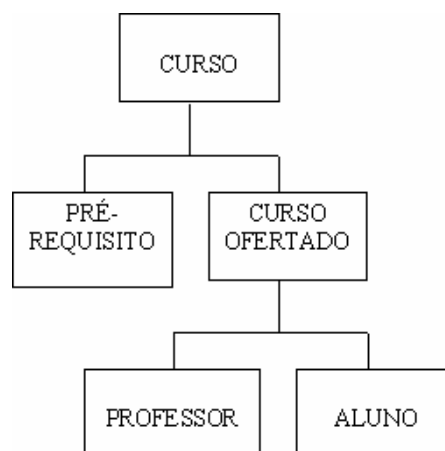


Fig. 1.2. 2 - Abordagem Hierárquica.

A definição dos dados na DDL deve ser feita em seqüência, ou seja:

- Define-se o banco de dados;
- Define-se o arquivo;
- Definem-se os campos desse arquivo;
- A cada arquivo hierarquicamente inferior precisa apontar o seu pai, relacionando os seus campos e assim por diante.

Produto Exemplo: IMS (Information Management System) da IBM

Sistemas de Redes

CODASYL (Conference on Data Systems Languages) – Comitê COBOL

DBTG (Data Base Task Group) – Produziu recomendações (1971) como:

- Padrão para DDL;
- Padrão para DML;
- Proposta de SGBD IDMS.

Em 1984 produziu-se um padrão de linguagens para BD em rede (NDL – Network Data Language). Proposta de padrão de DML

Estrutura do banco de redes:

- Uma área de dados;
- Uma área para ligações;
- É possível que um filho tenha mais de um pai. Exemplo: Cliente e Fornecedor estão ligados ao mesmo Estado. Na estrutura hierárquica isso não era possível;
- É preciso guardar a definição do registro e a definição de cada ligação (ambos precisam ser armazenados).

Exemplo:

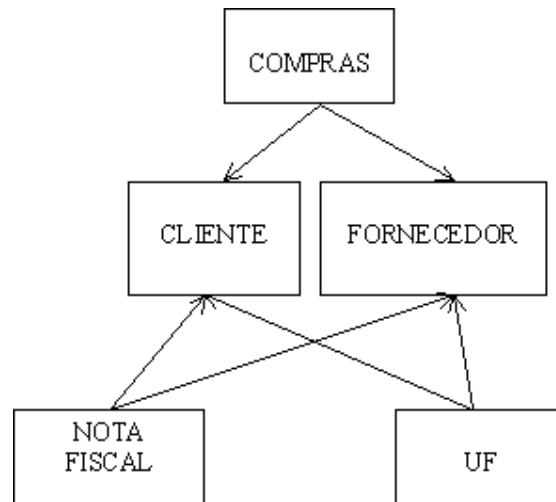


Fig. 1.2. 3 - Abordagem em Redes.

Observações:

- A linguagem padrão era o COBOL
- Para acessar dados é necessário obter as ligações e só então obter os dados.

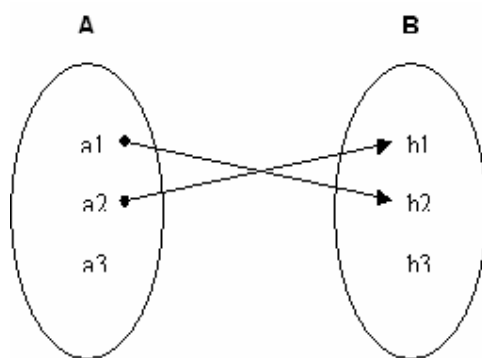
Exemplo de Produto: IDMS (Integrated Data Management System) da Cullinet Software. É a melhor implementação CODASYL (DBTG)

Sistemas Relacionais

Enquanto que as discussões e conceitos sobre bancos de dados não relacionais surgiram no meio empresarial, na indústria de software, ou seja, de forma empírica e experimental, o conceito de banco de dados relacional surgiu após discussões que ocorreram no meio acadêmico, pois os pesquisadores passaram a buscar um embasamento científico que pudesse dar sustentação teórica a teoria de banco de dados. A idéia mais óbvia que surgiu foi imaginar que um banco de dados é um lugar onde guardamos uma coleção de dados, ou seja, um banco de dados constitui um ou mais conjuntos de dados. Por conta disso, pensou-se em trabalhar a teoria de bancos de dados usando com base teórica a já consagrada teoria dos conjuntos.

A unidade básica de um banco de dados é o dado e, entende-se por dados ao conjunto de atributo que se precisa conhecer sobre alguma coisa, por exemplo, os dados de um aluno são a sua *matrícula*, seu *nome* e *sexo*. Portanto, esses dados nada mais são do que os atributos de aluno. Pensando em projeto lógico de banco de dados, teríamos uma estrutura *aluno* composta dos atributos *matrícula*, *nome* e *sexo*.

Agora, observemos as associações entre dois conjuntos, como o exemplo a seguir:



Efetivamente, os elementos de A que tem associação com B são (a1,a2) e, na Teoria dos conjuntos, diríamos que existe um subconjunto A' contendo (a1, a2) o qual encontra um subconjunto B' contendo (h1,h2), de modo que dizemos que A' é o domínio de A.

As associações desses conjuntos podem ser relacionadas como um conjunto $AB=\{a1b2, a2,b1\}$ e esse conjunto é chamado de Relação de A em B.

Se considerarmos que existe um conjunto de Matrículas, que encontra valores correspondentes de Nomes e Sexos, então teríamos o seguinte:

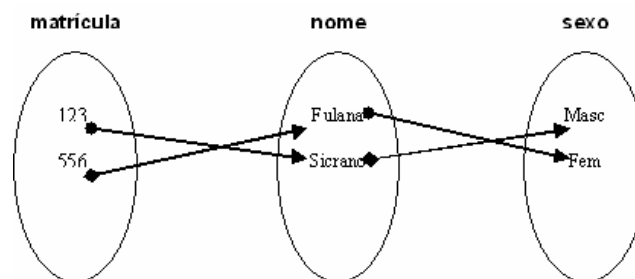


Fig. 1.2. 4 - Abordagem Relacional.

Fazendo analogia com os conjuntos do exemplo anterior, teríamos que:

- Domínio (matrícula)={123, 556}

- Domínio (nome) = {Fulana, Sicrano}
- Domínio (sexo) = {Masc, Fem}

Portanto, como matrícula, nome e sexo são os atributos dos nossos alunos, então, os valores possíveis de serem assumidos por cada um desses atributos é chamado de **domínio do atributo**. É possível que um determinado atributo possa ter um domínio mais amplo, como Salário > 1000. Mais tarde, esse domínio será conhecido como uma restrição do atributo, a qual deverá ser definido no banco via uma DDL.

Por analogia, ainda, podemos dizer que existe uma Relação, que chamaremos de $R=\{123\text{-Sicrano-Masc}, 556\text{-Fulana-Fem}\}$, mas, se observarmos bem, esses nada mais são que os diversos dados de cada Aluno. Portanto, Aluno representa uma **Relação**. Por isso, os bancos de dados que representam relações são conhecidos como **Bancos de Dados Relacionais**.

Se, genericamente, dissermos que existem t possibilidades de associações entre os valores dos atributos da relação Aluno, então, podemos dizer que existem múltiplas associações de valores de seus atributos, de modo que uma determinada associação pode ser uma ocorrência t -upla. Portanto, dizemos que os dados de cada um dos alunos da Relação Aluno é uma **tupla** de Aluno.

Por conseguinte, para que a Teoria de Bancos de Dados Relacionais tenha sustentação na Teoria dos conjuntos, dizemos que:

- As estruturas de dados que obtemos no nosso projeto lógico serão representados num banco de dados relacional como relações;
- Essas relações são constituídas de tuplas;
- As tuplas representam as associações dos valores dos atributos da relação;
- Os valores que cada atributo pode assumir correspondem ao seu domínio.

Alguns teóricos de bancos relacionais não admitem que se utilize para a manipulação de bancos relacionais a terminologia usada em sistemas de arquivos.

Quando os fabricantes de SGBDs partiram para implementar os conceitos definidos para bancos relacionais, não quiseram utilizar os conceitos matemáticos advindos da Teoria dos Conjuntos, preferiram utilizar conceitos mais óbvios para o leigo. A seguir apresentamos uma correspondência dos termos utilizados em bancos de dados relacionais com sistemas de arquivos:

Sistema de Arquivos	Modelo Relacional	Usuário (Produto)
Arquivo	Relação	Tabela
Registro	Tupla	Linha
Campo	Atributo	Coluna

Fig. 1.2. 5 - Comparação entre terminologias.

Podemos dizer que:

- Bancos de dados relacionais são baseados em relações ou coleções de relações;
- Bancos de dados relacionais são aqueles que contém coleções de tabelas;
- A base é a teoria relacional, a qual é baseada na teoria dos conjuntos;
- As operações relacionais são descritas na álgebra relacional;
- Os sistemas que permitem definição de tabelas mas não implementam as operações relacionais são chamados de tabulares;
- Os sistemas que permitem definição de tabelas e as operações relacionais básicas são chamados de minimamente relacionais;
- Os sistemas relacionais deveriam obedecer a todos os preceitos relacionais.

Características dos bancos de dados relacionais:

- Não há tuplas duplicadas;
- Não há ordenamento de tuplas;
- Não há ordenamento de atributos;
- Os valores dos atributos são atômicos, ou seja, não são permitidos atributos multivalorados. As relações estão na 1FN.
- Um banco relacional é uma coleção de relações normalizadas;
- As tuplas devem conter um atributo identificador ou combinação de atributos compondo uma chave primária;
- Os dados são manipulados por operações relacionais;
- Guardam independência lógica das aplicações: alterações na estrutura lógica do banco não obrigam manutenção na aplicação, desde que as alterações preservem os dados;
- Guardam independência física das aplicações: alterações ou remanejamento de espaço físico não obrigam a manutenção das aplicações;
- Guardam independência de integridade das aplicações, ou seja, uma alteração de valores de tabelas relacionadas podem ser definidas para refletirem automaticamente em outras tabelas. Ou seja, é possível definir no banco que caso seja excluída uma linha de uma tabela referenciada, o banco pode automaticamente excluir as tuplas correspondente a essa chave estrangeira em outra tabela;
- Todas as definições de dados e o próprio dicionário de dados ficarão logicamente fazendo parte de uma estrutura de dados chamada de catálogo e podem ser obtidas utilizando-se a mesma linguagem que é usada para fazer listar os dados de uma tabela. Em outras palavras, o SGBD relacional sempre tem um conjunto de tabelas (tabelas internas) que nada mais são que tabelas que contém as

definições de dados do banco de dados, as quais foram criadas por uma DDL.

Por uma questão de simplificação, ao invés de usar uma DDL para definição de dados e uma DML para a manipulação dos dados, a ANSI propôs que se pudesse usar uma única linguagem para ambos os propósitos. A seguir apresentaremos a sintaxe das sentenças SQL-ANSI que passaremos a utilizar para a definição de dados de um banco relacional. Faremos isso através de um exemplo.

Considere que o projeto lógico de um determinado banco de dados produziu o seguinte mapeamento:

Banco de dados: **RH**

Depto

Codep	NUM(6) NOT NULL
Nomedep	CHAR(20) NOT NULL

Empregado

Codemp	NUM (8)	(PK)
Nomemp	CHAR (40)	
Codep	NUM (6)	(FK)

Para implementar as estruturas de dados mapeadas é necessário que se emita sentenças que possibilitem ao SGBD a criação e manutenção dessas estruturas, como mostrado a seguir:

Para criar o banco de dados:

CREATE DATABASE RH;

Para poder utilizar o banco já criado:

OPEN DATABASE RH;

Para criar as tabelas:

```
CREATE TABLE DEPTO (  
    CODEP NUMERIC (8) NOT NULL,  
    NOMEDEP CHAR (40) NOT NULL,  
    PRIMARY KEY (CODEP));  
  
CREATE TABLE EMPREGADO (  
    CODEMP NUMERIC (8) NOT NULL,  
    NOMEMP CHAR (40) NOT NULL,  
    CODEP NUMERIC (6) NOT NULL,  
    PRIMARY KEY (CODEMP),  
    FOREIGN KEY (CODEP) REFERENCES DEPTO);
```

Caso desejemos eliminar uma tabela:

```
DROP TABLE DEPTO;
```

Se houvesse necessidade de modificar as características de um atributo:

```
ALTER TABLE DEPTO (  
    MODIFY NOMEDEP CHAR (50));
```

Se precisássemos acrescentar mais um atributo:

```
ALTER TABLE EMPREGADO (  
    ADD SALARIO DECIMAL (10));
```

Para criar uma chave secundária:

```
CREATE INDEX KEYNOME ON EMPREGADO (NOMEMP ASC);
```

Se não tivermos mais nada a fazer com o banco de dados, devemos fechá-lo:

```
CLOSE DATABASE RH;
```

ATIVIDADE

Responda as questões a seguir:

1. Como é constituída a Arquitetura ANSI/SPARC?
2. Qual a linguagem de manipulação de banco de dados é responsável pela descrição de esquemas?
3. Qual a linguagem de manipulação de banco de dados é usada para fazer acesso às instâncias de dados?
4. Cite duas características dos bancos de dados não relacionais?
5. Qual característica dos bancos de dados hierárquicos foi determinante no surgimento dos bancos de dados de redes?
6. Qual foi o fator determinante para o surgimento dos bancos de dados relacionais?