Unidade 2

UMA ARQUITETURA PARA SISTEMA DE BANCO DE DADOS

Nesse capitulo você verá uma proposta de arquitetura para um Sistema de Banco de Dados em três camadas, sendo uma para a Visão Externa, outra para a Visão Conceitual e uma terceira para a Visão Interna dos dados. Verá que tal arquitetura proporciona certa liberdade ou independência tanto na manutenção dos dados, quanto das aplicações. Aprenderá também quais os componentes fundamentais que um SGBD deve fornecer, tanto para a parte funcional quanto para o controle do armazenamento dos dados.

CAPITULO 2 – UMA ARQUITETURA PARA SISTEMA DE BANCO DE DADOS

2.1 OS TRÊS NÍVEIS DA ARQUITETURA

Conforme foi estudado no capitulo 1, uma das características importante proporcionada pela abordagem de banco de dados diz respeito à Independência de Dados, ou seja, a separação entre programas de aplicação e os dados manipulados por estes programas. Este isolamento entre programas e dados, proporcionou uma nova abordagem também no tratamento dos dados, que passaram a ser vistos como composto de dois elementos também isoláveis, ainda que intimamente relacionados. São eles o Modelo dos Dados e as Instâncias dos dados propriamente ditos. Em resumo, a abordagem de Banco de Dados conseguiu isolar três elementos interdependentes, mas que podem ser tratados separadamente: Os Programas (parte funcional), os Dados (ocorrências ou instâncias de valores em um determinado momento) e o Modelo dos Dados (Estrutura dos dados). (ELMASRI/NAVATHE, 2005, p.7)

O grupo de estudo de sistema de gerencia de banco de dados ANSI/SPARC propôs uma arquitetura que divide um ambiente de banco de dados em três níveis conhecidos como Nível Interno, Nível Externo e Nível Conceitual. Os Níveis Externo e Conceitual preocupam-se com o **modelo dos dados** que estão voltados para os usuários, enquanto o nível interno preocupa-se com a **implementação dos dados** que irão povoar os modelos de dados e estão voltados para a máquina (DATE, 2003).

Nível visual ou externo

É a forma como os Dados são vistos pelos usuários (individualmente). Estes usuários, tanto podem ser um programador de aplicações, como um usuário final. O DBA é um caso especial, pois terá de se interessar pelos níveis conceitual e interno também.

Qualquer tipo de usuário poderá atuar no nível externo, ou seja, poderá ter acesso ao sistema de banco de dados, respeitando-se é claro os limites de segurança impostos pelos mesmos. No entanto, cada usuário tem uma forma, uma linguagem à sua disposição para poder efetuar suas interações.

Os desenvolvedores, possivelmente utilizarão alguma linguagem de programação convencional (Java, C++ ou PLI) ou uma linguagem especifica de acordo com as particularidades de um sistema específico.

Já os usuários finais possivelmente irão interagir com o sistema de banco de dados através de aplicações on-line, elaborada a partir de formulários e menus, que possuem como forma de implementação alguma linguagem de consulta (provavelmente SQL). Não é comum que um usuário final tenha acesso as linguagens de programação utilizadas pelos desenvolvedores. O mais comum é que os desenvolvedores elaborem sistemas aplicativos, utilizando-se de alguma linguagem de programação e disponibilizem tais sistemas para uso dos usuários finais. Estes sistemas geralmente comporão um elenco de funções de interação com o banco de dados ao qual se referenciam. Tais funções incluirão ações tipo inserir, alterar, excluir ou simplesmente consultar os dados. Dessa forma, é a partir da execução desses sistemas aplicativos que os usuários finais terão acesso aos dados armazenados.

Para nossos propósitos, o que importa é sabermos que todas essas linguagens, tanto as utilizadas pelos desenvolvedores quanto as aplicações elaboradas para os usuários finais incluirão uma ***sublinguagem de dados*** - ou seja, um subconjunto de toda a linguagem, **voltado para os objetivos e operações do banco de dados**. Qualquer sublinguagem de dados é na verdade, uma combinação de pelo menos duas linguagens subordinadas: DDL – Data Definition Language que oferece recursos para declaração de todos os objetos do banco de dados e DML – Data Manipulation Language que oferece recursos para manipulação desses objetos.

As linguagens de definição dos dados (DDL) atuam na formatação do esquema ou estruturas dos dados enquanto as linguagens de manipulação de dados (DML) atuam nas instâncias ou ocorrências dos dados que povoam aquelas estruturas.

A sublinguagem de dados é embutida na *linguagem hospedeira* correspondente, que por sua vez proporciona os diversos recursos não específicos de banco de dados, tais como variáveis locais (temporais), operações computacionais, lógica *if-then-else* e assim por diante. A sublinguagem de dados mais comumente reconhecida pelos sistemas atuais é a SQL, que será estudada posteriormente. O ponto exato de fusão ou separação entre linguagem hospedeira e sublinguagem de dados é um tanto abstrato para usuário.

Analisando a comunidade de usuários de um banco de dados, percebemos interesses distintos de cada um sobre o complexo volume de dados armazenados. Cada usuário tem de acordo com a necessidade de informações que precisa visões diferentes sobre o mesmo banco de dados. Alem disso, algumas vezes a visão que o usuário tem sobre os dados que necessita é um tanto diferente da forma como estes dados estão verdadeiramente armazenados ou mesmo estruturados. Em termos de ANSI/SPARC, a visão de determinado usuário é uma **Visão Externa**. A Visão Externa é, portanto, o conteúdo do banco de dados como visto por determinado usuário, ou seja, para aquele usuário, a Visão Externa é o banco de dados (DATE, 2003).

Nível conceitual

É a visão global do grupo de usuários. A **Visão Conceitual** oculta dos usuários elementos do armazenamento físico e se concentra na descrição de entidades, tipos de dados e restrições, por exemplo. A **Visão Conceitual** é a representação de todo o conteúdo de informações do banco de dados, que não necessariamente será igual nem ao armazenamento físico ou a Visão Externa que os usuários têm desses dados.

Resumidamente, podemos dizer que a **Visão Conceitual** é a visão dos dados “como realmente são”, e não como os usuários são forçados a vê-los devido às restrições de linguagem ou hardware. As definições no esquema conceitual devem incluir aspectos de segurança e integridade, e não somente a descrição de todos os dados operacionais.

A definição do esquema conceitual se dá a partir de uma linguagem especifica chamada DDL conceitual. As definições feitas a nível conceitual se quiserem atender aos critérios de independência de dados propostos devem ser imunes à representação física e a técnica de acesso. Deverá ser tratado no nível conceitual somente as definições do conteúdo da informação.

As definições no esquema conceitual incluem elementos adicionais como, por exemplo, restrições de integridade e segurança (ELMASRI/NAVATHE, 2005 e DATE, 2003).

Nível interno ou físico

É a forma como os dados estão armazenados fisicamente. É uma pequena representação de todo o banco de dados; São as múltiplas ocorrências de tipos múltiplos de **registros internos**. O **registro interno** equivale ao **registro armazenado**. A Visão Interna é descrita por meio do **esquema interno**, que não só define os vários registros armazenados como também especifica os índices que existem, como os campos armazenados são representados e a seqüência física dos registros armazenados.

O Nível Interno ainda não se refere ao armazenamento propriamente dito de um dado, mas sim de como os dados deverão estar armazenados no nível físico. Falando de outra maneira, a **Visão Interna** supõe que exista no meio de armazenamento um espaço linear infinito, porém os detalhes de como esse espaço de endereços é mapeado no meio de armazenamento físico, se os bits equivalentes aos dados a serem armazenados estarão todos efetivamente em um único espaço ou distribuídos ao longo do dispositivo de armazenamento, são muito específicos e foram deliberadamente omitidos da arquitetura geral.

A maioria dos SGBDs não separa os três níveis completamente, pois incluem elementos de nível físico no esquema conceitual. Aqueles SGBDs que oferecem possibilidade de múltiplas visões dos usuários incluem os múltiplos esquemas externos no mesmo modelo que descreve o esquema conceitual (ELMASRI/NAVATHE, 2005, p.23)

Mapeamentos

Sempre que um usuário solicita um determinado dado armazenado, ele irá invocar um esquema externo, que por sua vez ira analisar o esquema conceitual correspondente aquele esquema externo, que então irá fazer a solicitação no esquema interno a fim de obter a informação solicitada pelo usuário. Uma vez obtido o dado no nível interno, este deve ser então transformado para adaptar-se a Visão Externa do usuário. O processo de transformação é denominado **mapeamento** (DATE, 2003).

São disponibilizados dois níveis de mapeamentos: Um mapeamento entre os níveis ***Conceitual/Interno*** e um entre os níveis **Externo/Conceitual**.

O Mapeamento Conceitual/Interno trata do processo de acoplamento entre os níveis conceitual e os dados armazenados; é responsável por definir como os registros e campos modelados no nível conceitual são representados no nível interno. Uma modificação na definição da estrutura armazenada deverá ocasionar mudanças no mapeamento conceitual/interno, de forma que o esquema conceitual permaneça invariável. Essas ações são de responsabilidade do DBA ou do próprio SGBD e devem ser transparentes ao usuário final, devendo ser isoladas abaixo do nível conceitual a fim de preservar a independência física.

O Mapeamento Conceitual/Externo responde pela correspondência entre uma determinada Visão Externa e a Visão Conceitual. Os campos podem ter tipos de dados diferentes, as denominações de campo e registro podem ser modificadas. Campos conceituais podem ser combinados em um único campo externo.

É através dos conceitual/interno e externo/conceitual que se tem condições de garantir a independência de dados física e lógica respectivamente.

2.2 INDEPENDÊNCIA DE DADOS

Os sistemas que não são suportados por Banco de Dados, costumam ser Dependentes de Dados. Nesses sistemas, "a maneira como os dados são representados fisicamente no meio de armazenamento secundário, bem como a técnica usada para obter acesso a eles são ambas determinadas pelos requisitos da aplicação que está sendo considerada e, acima de tudo, que *o* **conhecimento dessa representação física e dessas técnicas de acesso está embutido no código da aplicação***"* (DATE, 2003, p.18).

Por outro lado, a Independência de Dados pode ser definida como a "Imunidade das aplicações à estrutura de armazenamento e à estratégia de acesso", ou seja, diminuir as alterações de programas devido a modificações nos dados do banco de dados. As modificações nos dados não se referem às atualizações temporais dos valores dos itens de dados e sim quanto ao tamanho, formato, precisão, escala, denominação padrão, de itens de dados.

Para entendermos melhor a Independência de Dados, precisamos compreender dois conceitos muito importantes: **Dado** e **Modelo de Dados**.

A palavra **dado** significa "dar" em latim. Os **dados** são a representação física de um fato. Por ser um fato, é sempre uma proposição verdadeira. Um Banco de Dados é uma coleção de dados, agrupados segundo alguma relação lógica entre eles, como por exemplo, o conjunto dos dados dos alunos, o conjunto dos dados dos materiais, etc. Falando em termos computacionais, dizemos que os dados são vistos como cadeias de bits armazenados. O conceito de dado está relacionado também a três conceitos básicos: **campo armazenado, registro armazenado e arquivo armazenado***:*

- **Campo armazenado** → Menor unidade de dado armazenado. Representa o valor atribuído a uma determinada característica de uma entidade do mundo real. Por exemplo: "Masculino" é o valor atribuído ao campo "sexo" de uma entidade "empregado". Observe que no exemplo, dois elementos parecem se sobrepor, mas na verdade cumprem papeis bem distintos. São eles: **tipo** e **ocorrência**. O **tipo** "sexo" é diferente da **ocorrência** "masculino". O **tipo** refere-se a características que a entidade deve ou pode possuir já a ocorrência, refere-se ao efetivo valor atribuído a esta característica.

- **Registro armazenado** → Coleção de campos armazenados relacionados entre si. Aqui também ocorre a mesma distinção entre **tipo** e **ocorrência**. Semelhante ao que ocorre com o campo armazenado, um registro, antes de ser povoado com valores (ocorrências) possui um "layout", uma estrutura que define os campos (características) atribuídos à entidade que representa. Este layout, estrutura, define o **tipo** de registro. No entanto, quando atribuímos valores as diversas características, aos diversos campos, temos as **ocorrências** dos registros.

- **Arquivo armazenado** → Coleção de todas as **ocorrências**, de um único **tipo** de **registro armazenado**.

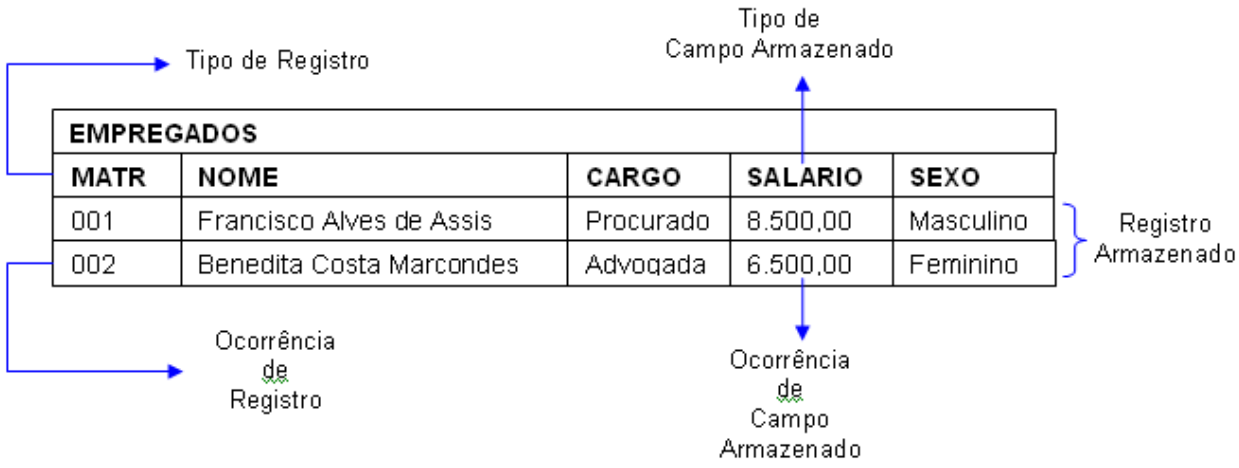

FIGURA2.1 Campo, Registro e Arquivo armazenado

Em sistemas que não utilizam as técnicas de Banco de Dados, o registro lógico (aquele visto por uma aplicação) corresponde exatamente ao registro armazenado. Já nos sistemas baseados em Banco de Dados, o registro armazenado pode ser bem distinto do que a aplicação enxerga, pois o DBA pode fazer modificações na representação dos dados armazenados (campos, registros e arquivos) enquanto os dados vistos pela aplicação não se alteram. Por exemplo, o campo SALARIO poderia ser armazenado em **binário** para economizar espaço de armazenamento ao passo que uma aplicação poderia vê-lo como **string.** Posteriormente, o DBA poderia fazer outra modificação nessa representação para **decimal** e continuaria sendo visto como **string**. (DATE, 2003, p.20,21)

Um **modelo de dados** é uma definição abstrata, autônoma e lógica dos objetos, operadores e outros elementos que juntos constituem a **maquina abstrata** com os quais os usuários interagem. Os **objetos** nos permitem modelar a **estrutura dos dados** enquanto os **operadores** nos permitem modelar o seu ***comportamento***.

Uma vez conceituado campo, registro e arquivo armazenado, podemos então voltar ao tratar de Independência de Dados. Como foi dito anteriormente, Independência de Dados é a imunidade da aplicação à estrutura de armazenamento e à estratégia de acesso. Isso quer dizer que modificações efetuadas no layout ou mesmo na forma de armazenamento devem ser tratadas da maneira mais transparente possível pela aplicação. Existem dois níveis de Independência de Dados (SILBERSCHATZ; KORTH; SUDARSHAN; 1999, p.6)

- Independência Física de dados: É a habilidade de modificar o esquema físico sem a necessidade de reescrever os programas aplicativos. As modificações no nível físico são ocasionalmente necessárias para melhorar o desempenho.

  Ex1. Um arquivo armazenado pode estar totalmente contido em um dispositivo de disco, ou pode ser fragmentado.

  Ex2. Os dados numéricos podem ser armazenados de maneira diferente para economizar espaço, como por exemplo, decimal compactado.

Ex3. Caracteres podem ser armazenados por codificações distintas, por exemplo: EBCDIC, ASCII.

- Independência Lógica de dados: É a habilidade de modificar o esquema lógico sem necessidade de reescrever os programas aplicativos. As modificações no nível lógico são necessárias quando a estrutura lógica do banco de dados é alterada.

Ex1. Podemos acrescentar novas colunas na estrutura de um arquivo

Ex2. Modificar a posição das colunas dentro do leiaute, sem que se afetem os programas que modifiquem ou leiam esta tabela.

Certamente que, esta liberdade de se modificar os esquemas, tanto conceitual quanto físico só será possível porque se faz a separação entre o **Modelo de Dados** e o **Dado** que o povoa. A estrutura do dado está isolada do dado fisicamente armazenado. Dessa forma, tem-se certa liberdade para modificar um ou outro, sem afetar a aplicação que utiliza este dado.

2.3 ESTRUTURA DO SGBD

O Sistema de Gerenciamento de Banco de Dados é o software que manipula todos os acessos ao banco de dados. Alguns autores se referem ao SGBD como um Sistema de Gerenciamento de Banco de Dados de Propósito Geral, contrastando com os sistemas aplicativos que desenvolvemos para os usuários finais que são enquadrados como sendo Sistemas de Gerenciamento de Banco de Dados de Propósito Específico. Mas no dia-a-dia é comum que se use o termo Sistemas Aplicativos para denominar os SGBDs de propósito específico e que se use SGBD para denominar os SGBDs de propósito geral. Nesse material, sempre que nos referirmos a palavra SGBD estaremos tratando dos SGBDs de propósito geral. Estes são um conjunto de programas desenhados para desempenhar determinadas funções a fim de atingir objetivos específicos envolvendo a formação e utilização de banco de dados (DATE, 2003).

O Sistema de Gerenciamento de Banco de Dados do Inglês *Data Base Managment System* é o software que manipula todos os acessos ao banco de dados. Esses acessos podem ser sintetizados nos passos abaixo:

i. O usuário emite uma solicitação de acesso, usando uma sublinguagem específica de dados.
ii. O DBMS intercepta a solicitação e a analisa.
iii. O DBMS, por sua vez, inspeciona os esquemas externos para aquele usuário, o mapeamento externo/conceitual, o esquema conceitual, o mapeamento conceitual/interno e a definição da estrutura de armazenamento.
iv. O DBMS executa as operações necessárias no banco de dados armazenado.

Esta estruturação está um tanto simplificada para que você possa entender, em que momento o SGBD intercepta uma solicitação vinda do mundo externo (programas aplicativos, *queries*) e administra o processo até a efetiva conclusão da solicitação, seja ela uma leitura, gravação, eliminação, etc, dos dados armazenados.

Para poder realizar suas tarefas e atender as solicitações vindas do mundo externo o SGBD deve ser munido de um conjunto de funções. Tais funções devem dar suporte aos requisitos definidos abaixo (SILBERSCHATZ; KORTH; SUDARSHAN; 1999, p.16, p.17).

- **Componentes Funcionais para Processamento de Consulta**:

a) *Compilador DML:* Traduz comandos DML em instruções de baixo nível inteligíveis ao Processador de Consultas. Além disso, o compilador, também tem a responsabilidade de analisar a requisição do usuário e definir a melhor estratégia de acesso ao dado.

b) *Pré-compilador de DML:* Converte comandos da DML[1] embutida em um aplicativo para chamadas de procedimento normal na linguagem hospedeira. O pré-compilador precisa interagir com o Compilador DML de modo a gerar o código apropriado. Possivelmente uma linguagem hospedeira pode ser qualquer linguagem que forneça recursos para a elaboração de sistemas aplicativos.

c) *Interpretador de DDL:* Converte comandos da DDL[2] em um conjunto de tabelas contendo *metadados* ou "dados sobre dados". Comandos DDL são os que atuam construindo os esquemas do banco de dados.

d) *Processador de Consultas:* Executam instruções de baixo nível geradas pelo compilador de DML.

- **Componentes Funcionais para Administração de Memória**:

a) *Gerenciamento de Autorizações e Integridade:* Analisa o cumprimento das Regras de Integridade e a permissão do usuário de acesso ao Banco de Dados.

b) *Gerenciamento de Transações:* Garante a consistência do banco em processos de atualizações bem como questões relacionadas a processamentos concorrentes. Uma transação deve ter a capacidade de garantir as propriedades básicas de Atomicidade, Consistência, Isolamento e Durabilidade (ACID). A Atomicidade é o principio pelo qual a execução de uma transação ou é totalmente concluída ou deverá ser abortada. A Consistência visa garantir que os dados permaneçam íntegros após a execução de uma transação, ou seja, nenhuma Restrição de Integridade controlada pelo SGBD poder ser violada pela execução de uma transação. O Isolamento é a propriedade pela qual, múltiplas transações podem estar sendo executadas de maneira concorrente sem que uma interfira nas outras e a Durabilidade garante que a partir do ponto de *COMMIT* de uma transação (ou seja, ponto de confirmação da transação), as modificações realizadas por esta transação serão permanentes ou duradouras no banco de dados.

c) *Administração de Arquivos:* Gerencia a alocação de espaço de armazenamento e as estruturas de dados usadas para o armazenamento.

d) *Administração de Buffer:* Responsável pela transferência dos dados do disco/memória e pela decisão de quais dados colocar na memória *Cache*.

Diversas **Estruturas de Dados** são requeridas como parte da implementação do sistema físico, incluído:

a) *Arquivo de Dados:* Armazena o próprio banco de dados.

b) *Dicionário de Dados:* Armazena informações sobre os dados do banco de dados; Os dados residentes no Dicionário de Dados são também chamadas de metadados (no caso do Banco de Dados Relacional, dentro de tabelas) e contem informações sobre os esquemas suportados pelo SGBD, restrições de integridade, formato dos dados suportados pelo SGBD, restrições de seguranças, chaves de acesso.

c) *Índices:* Permite o acesso mais rápido aos dados. São estruturas de dados adicionais utilizadas para agilizar o processo de leitura a determinado dado armazenado. Informações mais detalhadas sobre índices serão vistos na unidade 3.

d) *Estatísticas:* Armazenam informações sobre o banco de dados e é usado pelo seletor de estratégias de acesso. Estas informações são utilizadas pelo processador de consultas para escolher a melhor estratégia para execução da consulta. Informações sobre o volume de dados armazenados, a freqüência com que determinado dado é consultado, o índice de repetição de determinado valor dentro de determinada coluna, são alguns dos elementos que ficam armazenados nas estatísticas.

A figura 2.1 demonstra a relação que existe entre cada um dos componentes da estrutura do SGBD.

[1]DML - Data Manipulation Language. Linguagem utilizada para obtenção de dados armazenados nos bancos de dados, inserções e remoções.

[2]DDL - Data Definition Language - Linguagem utilizada para criação e manutenção de bancos de dados e tabelas.

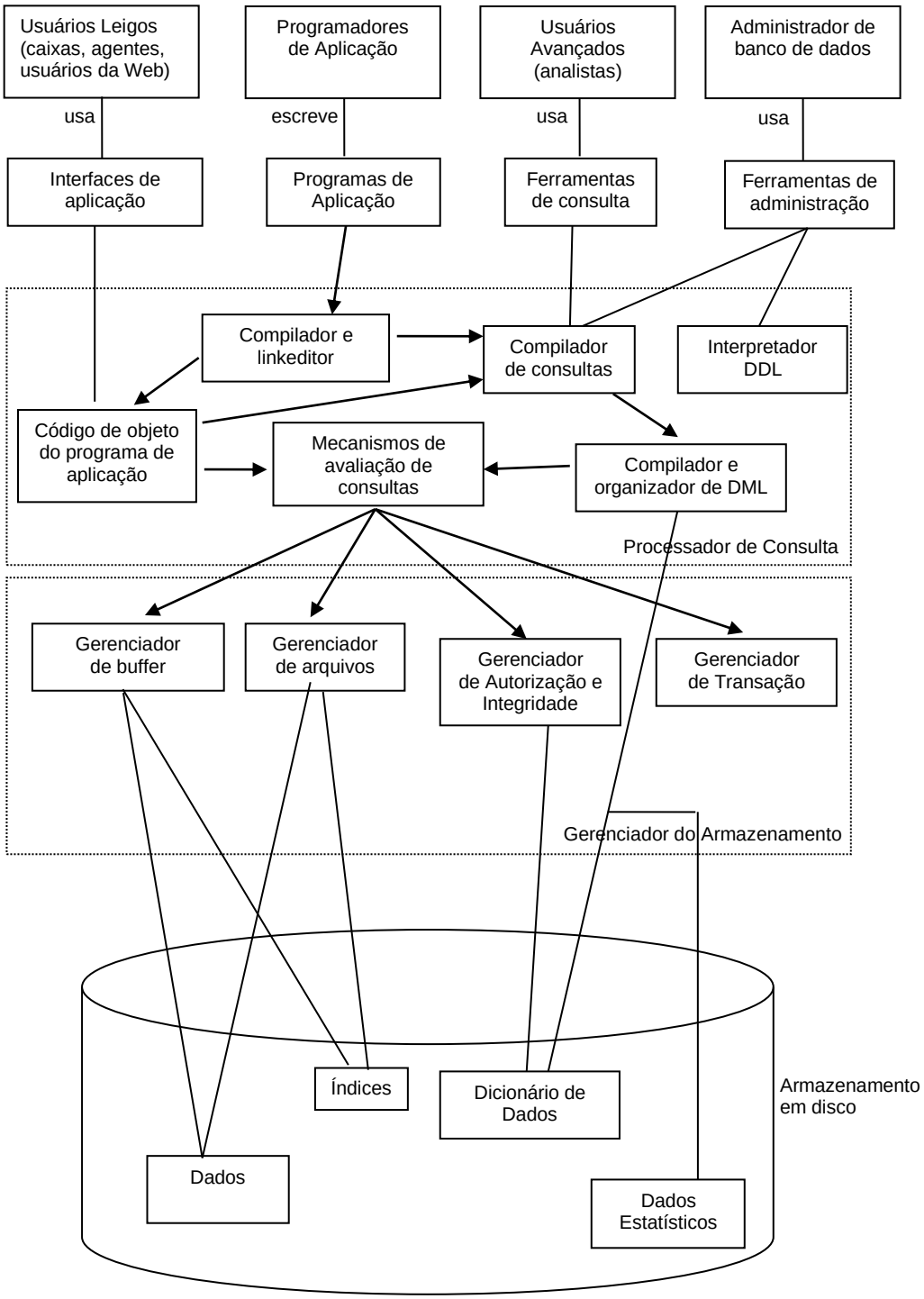

FIGURA2.1 *Estrutura Geral do Sistema*
Fonte: Silberschatz, Korth e Sudarshan, 2006

BIBLIOGRAFIA

DATE, C.J. **Introdução a Sistemas de Banco de Dados.** 8ed. Rio de Janeiro: Elsevier, 2003.

ELMASRI, R., NAVATHE, S. B. **Sistemas de Banco de Dados.** 4ed. São Paulo: Adisson Wesley, 2005.

KORTH, H. F. & SILBERSCHATZ, A. **Sistema de Bancos de Dados.** 3ed. São Paulo: Makron Books, 1999.

KORTH, H. F. & SILBERSCHATZ, A. **Sistema de Bancos de Dados.** 5ed. Rio de Janeiro: Elsevier, 2006.