



INF0367 – Banco de Dados 1

Objetivos: Apresentar os principais conceitos de Bancos de Dados, com ênfase na Modelagem de Sistemas e nos Projetos Conceitual, Lógico e Físico de Bancos de Dados.

Ementa: Fundamentos de Banco de Dados; Modelo de Dados; Linguagem de definição e manipulação de banco de dados (SQL-92); Projeto Conceitual de Banco de Dados; Projeto Lógico de Banco de Dados; Projeto Físico de Banco de Dados.

Professor: Paulo Cesar Azevedo Teixeira.

Conteúdo Programático

Unidade 1. Fundamentos de Banco de Dados

Unidade 2. Modelos de Dados

Unidade 3. Linguagem de Definição e Manipulação de Bancos de Dados

Unidade 4. Controles Operacionais de Bancos de Dados

Unidade 5. Administração de Bancos de Dados

Unidade 6. Projeto Conceitual de Bancos de Dados

Unidade 7. Projeto Lógico de Bancos de Dados

Unidade 8. Projeto Físico de Bancos de Dados

Bibliografia

- Date, C. J. - Introdução a Sistemas de Bancos de Dados - 7.ed. - São Paulo: Ed. Campus, 2000.
- Korth, H. F. , Silberschatz, A. - Sistemas de Bancos de Dados - 3. ed. - São Paulo - Makron 1991.
- Ramez Elmasri, Shamkant B. Navathe - Fundamentals of Database Systems - 3.ed. - Addison Wesley: 2000.

1. Fundamentos de Banco de Dados

Dados e Informações

Organizações ou Empresas são um conjunto de processos relacionados que consomem recursos e apresentam resultados. Os recursos consumidos podem ser classificados como Recursos Humanos, Recursos Materiais, Recursos Financeiros e Recursos da Informação. Os processos internos da organização, quando estabelecidos, trocam recursos entre si através da forma mais comum de relacionamento: *a troca de informações*. Algumas definições:

a) Dado

- Estímulos captados pelos sentidos humanos;
- Símbolos gráficos ou sonoros;
- Ocorrências registradas na memória humana, no papel, em arquivos, em bancos de dados;
- Indica uma situação (status);
- Define o conteúdo de um campo; e
- Não transmite conhecimento.

b) Informação

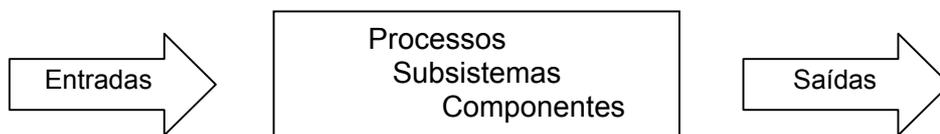
- Dados processados e com significado para o receptor;
- Transmite conhecimento e tem valor real ou percebido;
- É um recurso da organização, como o seu patrimônio e equipamentos;
- Características e requisitos próprios, que determinam seu uso na organização; e
- Apresentação diferenciada (gráficos, vídeos, textos, etc.)

A informação tem custos de produção (coleta, organização, transformação, armazenamento e manutenção) e custos de distribuição. Conseqüentemente terá um preço estabelecido em função do valor agregado (para quem a utilizará – o cliente) e da escala de utilização. A informação também possui os seguintes requisitos:

- Conteúdo → o que é transmitido
- Formato → alfabético; numérico; imagem
- Quantidade → em termos de volume e periodicidade
- Qualidade → que é uma composição dos seguintes fatores:
 - . Adequabilidade → conteúdo e formato compatíveis com a natureza da decisão
 - . Confiabilidade → garantia de origem/ precisão
 - . Integridade → precisão de conteúdo
 - . Acessibilidade → facilidade de obtenção
 - . Oportunidade → disponível no momento, local e forma adequadas
 - . Clareza → fácil entendimento.

c) Sistemas

Os sistemas podem ser definidos como um conjunto de partes interdependentes, ou um todo organizado, ou partes que interagem formando um todo unitário e complexo. Podemos ainda dizer que uma sistema é qualquer conjunto de componentes e processos por ele executados, que visam transformar determinadas entradas e saídas.



d) Sistemas de Informação

São conjuntos de procedimentos que visam captar o que acontece na organização, apresentando de forma sucinta e adequada a cada nível, as informações que lhe cabem e são necessárias, visando subsidiar o processo decisório. Os componentes típicos de qualquer sistema de informação são: pessoas, procedimentos, metodologias, equipamentos e software.

e) Tecnologia da Informação

Conjunto de hardware e software que desempenha tarefa de processamento de informações, fazendo parte do sistema de informação das organizações, como meio para alcançar seus objetivos. O papel da Tecnologia da Informação é dar suporte a informação em todo o seu ciclo de vida, compreendendo a obtenção, a transmissão, o processamento, o armazenamento e a exibição da informação.

Definição de Banco de Dados

Um banco de dados é uma coleção de dados interrelacionados, representando informações sobre um domínio específico. Estes dados são fatos que podem ser armazenados e tem significado implícito. Como exemplo podemos citar listas telefônicas, o controle do acervo de uma biblioteca e o sistema de controle de recursos humanos de uma empresa.

A definição de banco de dados apresentada é bastante abrangente; por exemplo, podemos considerar a coleção de palavras que compõem este texto como dados interrelacionados e, então, constituírem um banco de dados. Contudo, o uso comum do termo banco de dados é normalmente mais restrito. Um banco de dados tem as seguintes propriedades implícitas:

- Um banco de dados representa alguns dos aspectos do mundo real, algumas vezes chamado de mini-mundo. As mudanças neste minimundo são refletidas no banco de dados.
- Um banco de dados é uma coleção lógica e coerente de dados com algum significado. Uma coleção aleatória de dados não pode, corretamente, referir-se a um banco de dados.
- Um banco de dados é projetado, construído e povoado com dados para um propósito específico. Ele tem um grupo de usuários e algumas aplicações pré-concebidas nas quais estes usuários estão interessados.

Em outras palavras, um banco de dados possui alguma fonte de onde os dados são derivados, algum grau de interação com eventos no mundo real e uma clientela que está ativamente interessada no conteúdo do banco de dados. Desta forma, um banco de dados pode ser definido como sendo o componente da Tecnologia da Informação cuja estrutura e o comportamento refletem precisamente a estrutura e o comportamento do mini-mundo a ele correspondente; tem por finalidade armazenar, de forma persistente e consistente, dados sobre fatos e eventos ocorridos no mini-mundo, visando atender a um ou mais Sistemas de Informação.

Um banco de dados pode ser de qualquer tamanho e de complexidade variável. Por exemplo, uma lista de nomes e endereços pode conter algumas centenas de registros com uma estrutura simples. Por outro lado, o catálogo de uma grande biblioteca pode conter milhões de fichas armazenadas em diferentes categorias – por nome do autor, por assunto, por título do livro – e cada categoria organizada em ordem alfabética.

Um banco de dados pode ser criado e mantido manualmente ou pode ser computadorizado. O controle de fichas de uma biblioteca é um bom exemplo de um banco de dados criado e mantido manualmente. Um banco de dados computadorizado pode ser criado e mantido tanto por um grupo de programas escritos especificamente para este fim ou por um Sistema Gerenciador de Banco de Dados (SGBD). Um SGBD permite aos usuários criarem e manipularem bancos de dados de propósito geral. O conjunto formado por um banco de dados e mais as aplicações que o manipulam é chamado de “Sistema de Banco de Dados”.

Visões do Banco de Dados

A – VISÃO INTERNA – É aquela vista pelo responsável pela manutenção e desenvolvimento do SGBD. Existe a preocupação com a forma de recuperação e manipulação dos dados dentro do Banco de Dados.

B – VISÃO CONCEITUAL – É aquela vista pelo analista de desenvolvimento e pelo administrador das bases de dados. Existe a preocupação na definição de normas e procedimentos para a manipulação dos dados, para garantir a sua segurança e confiabilidade, o desenvolvimento de programas aplicativos e a definição no banco de dados de novos arquivos e campos. Nesta visão existem duas linguagens de operação que são:

1. *Linguagem de Definição de Dados (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);
2. *Linguagem de Manipulação dos Dados (DML)* – linguagem que define os comandos de manipulação e operação dos dados (comandos de consulta e atualização dos dados dos arquivos).

C – VISÃO EXTERNA – É aquela vista pelo usuário que opera os sistemas aplicativos, através de interfaces (programas), buscando o atendimento de suas necessidades.

Vantagens do Banco de Dados em relação à arquitetura tradicional

SISTEMA TRADICIONAL - São aqueles em que os dados do sistema estão armazenados fisicamente separados um do outro. O acesso é feito pelos programas de aplicação, associando o nome externo dos arquivos e definindo todo o registro independente da utilização dos campos.

SISTEMA DE BANCO DE DADOS - É aquele em que os dados são definidos para o S.G.B.D., através da DDL (linguagem de definição de dados). Fisicamente estão armazenados em um único local, sendo o acesso realizado apenas através do S.G.B.D. Nos programas de aplicação, é necessário apenas definir os campos que serão utilizados pelo programa.

VANTAGENS DO BANCO DE DADOS

1 - **REDUÇÃO OU ELIMINAÇÃO DE REDUNDÂNCIAS** - Possibilita a eliminação de dados privativos de cada sistema. Os dados, que eventualmente são comuns a mais de um sistema, são compartilhados por eles, permitindo o acesso a uma única informação sendo consultada por vários sistemas.

2 - **ELIMINAÇÃO DE INCONSISTÊNCIAS** - Através do armazenamento da informação em um único local com acesso descentralizado e, sendo compartilhada à vários sistemas, os usuários estarão utilizando uma informação confiável. A inconsistência ocorre quando um mesmo campo tem valores diferentes em sistemas diferentes. Exemplo, o estado civil de uma pessoa é solteiro em um sistema e casado em outro. Isto ocorre porque esta pessoa atualizou o campo em um sistema e não o atualizou em outro. Quando o dado é armazenado em um único local e compartilhado pelos sistemas, este problema não ocorre.

3 - **COMPARTILHAMENTO DOS DADOS** - Permite a utilização simultânea e segura de um dado, por mais de uma aplicação ou usuário, independente da operação que esteja sendo realizada. Deve ser observada apenas o processo de atualização concorrente, para não gerar erros de processamento (atualizar simultaneamente o mesmo campo do mesmo registro). Os aplicativos são por natureza multiusuário.

4 - RESTRIÇÕES DE SEGURANÇA - Define para cada usuário o nível de acesso a ele concedido (leitura, leitura e gravação ou sem acesso) ao arquivo e/ou campo. Este recurso impede que pessoas não autorizadas utilizem ou atualizem um determinado arquivo ou campo.

5 - PADRONIZAÇÃO DOS DADOS - Permite que os campos armazenados na base de dados sejam padronizados segundo um determinado formato de armazenamento (padronização de tabela, conteúdo de campos, etc.) e ao nome de variáveis seguindo critérios padrões preestabelecidos pela empresa. Ex. Para o campo "Sexo" somente será permitido armazenamento dos conteúdos "M" ou "F".

6 - MANUTENÇÃO DE INTEGRIDADE - Exige que o conteúdo dos dados armazenados no Banco de Dados possuam valores coerentes ao objetivo do campo, não permitindo que valores absurdos sejam cadastrados. Exemplo: Um funcionário que faça no mês 500 horas extras, ou um aluno que tenha nascido no ano de 1860.

7 - EVITAR NECESSIDADES CONFLITANTES – Prova a capacidade que o administrador de Banco de Dados deve ter para solucionar "prioridades sempre altas" de todos os sistemas, tendo ele que avaliar a real necessidade de cada sistema para a empresa para priorizar a sua implantação.

8 - INDEPENDÊNCIA DOS DADOS - Representa a forma física de armazenamento dos dados no Banco de Dados e a recuperação das informações pelos programas de aplicação. Esta recuperação deverá ser totalmente independente da maneira com que os dados estão fisicamente armazenados. Quando um programa retira ou inclui dados o SGBD compacta-os para que haja um menor consumo de espaço no disco. Este conhecimento do formato de armazenamento do campo é totalmente transparente para o usuário. A independência dos dados permite os seguintes recursos:

a - Os programas de aplicação definem apenas os campos que serão utilizados independente da estrutura interna dos arquivos

b - Quando há inclusão de novos campos no arquivo, será feita manutenção apenas nos programas que utilizam esses campos, não sendo necessário mexer nos demais programas.

Classificações de Banco de Dados

Vários critérios são utilizados para classificar os bancos de dados. O *primeiro critério* refere-se ao modelo de dados implementado; representa a estrutura física no qual o armazenamento dos dados foi projetado. O modelo identifica a estrutura interna de recuperação e armazenamento dos dados.

- **Bancos de Dados Convencionais** utilizam modelos lógicos baseados em registros. O banco de dados é estruturado em registros de formatos fixos de diversos tipos, cada tipo de registro tem sua coleção de atributos e há linguagens para expressar consultas e atualizações no banco de dados:
 - *Hierárquico* → os dados e relacionamentos são representados por registros e ligações, respectivamente. Os registros são organizados como coleções arbitrárias de árvores. O esquema hierárquico é mostrado como um diagrama hierárquico. Um detalhe importante a ser observado neste modelo é que os relacionamentos são do tipo pai-filho. Desta forma, um relacionamento N:N (muitos para muitos) não podem ser representados diretamente, uma vez que filhos só podem relacionar-se com um pai. Uma das maneiras de resolver este problema é repetir o registro em mais de um relacionamento.
 - *de Rede* → os dados são representados por coleções de registros, podendo incluir atributos compostos (vetores) e os relacionamentos por elos. Os registros são compostos por itens de dados reais ou virtuais.
 - *Relacional* → tanto os dados como os relacionamentos são representados por tabelas cada uma com suas colunas específicas. Este modelo de dados possui um fundamento matemático

sólido e prescindir de estruturas de índice eficientes e hardware adequado para alcançar desempenho viável em situações práticas.

- Outros Modelos
 - Orientado a Objetos → superficialmente podemos dizer que este modelo corresponde a organização de sistemas como uma coleção de objetos que integram estruturas de dados e comportamento. Os objetos são abstrações de dados do mundo real, com uma interface de nomes de operações e um estado local que permanece oculto. Um objeto tem um estado interno descrito por atributos que podem apenas ser acessados ou modificados através de operações definidas pelo criador do objeto. Um objeto individual é chamado de instância ou ocorrência de objeto.
 - Semântico → propõe a representação em banco de dados da semântica extraída diretamente do modelo conceitual, principalmente E/R; foram realizadas algumas tentativas de operar sobre um BD semântico, através de uma linguagem funcional denominada DATAPLEX.

Os dois tipos de modelos de dados usados em vários SGBD comerciais são o Modelo Relacional e o Modelo Orientado a Objetos. Muitas aplicações ainda rodam em sistemas de banco de dados baseados em modelos hierárquicos e de redes.

Os modelos de dados orientados a objetos são mais adequados ao tratamento de objetos complexos (textos, gráficos, imagens) e dinâmicos (programas e simulações). Estes modelos possuem uma maior naturalidade conceitual e estão em consonância com as tendências em linguagens de programação e engenharia de software. O casamento entre as linguagens de programação e banco de dados é um dos problemas que estão sendo tratados de forma mais adequada no contexto de orientação a objetos.

Os SGBD relacionais estão evoluindo continuamente e, em particular, tem incorporado muitos conceitos que foram desenvolvidos em bancos de dados orientados a objeto. Isto levou a uma nova classe de SGBD que vêm sendo chamados de SGBD objeto-relacional. Podemos, então, categorizar os bancos de dados de acordo com os modelos de dados da seguinte maneira: relacionais, orientados a objeto, objeto-relacionais, hierárquicos, de redes e outros.

O *segundo critério* utilizado para classificar bancos de dados é o número de usuários suportados pelo sistema. Sistemas *mono-usuários* suportam somente um usuário por vez e são mais comumente encontrados em computadores pessoais. Sistemas *multiusuários* suportam vários usuários concorrentemente.

O *terceiro critério* é o número de máquinas no qual o banco de dados está distribuído. O banco de dados é dito *centralizado* quando os dados estiverem armazenados em um único computador. Um banco de dados centralizado pode ser acessado por vários usuários mas o SGBD e os dados residem totalmente em um único computador. Um banco de dados distribuído pode ter a base de dados e o software de gerenciamento distribuídos em várias máquinas, conectadas por uma rede de computadores. Bancos de dados homogêneos usam o mesmo software de gerenciamento em várias máquinas. Uma tendência recente é o desenvolvimento de software que possa acessar bases de dados autônomas armazenadas sob SGBD heterogêneos. Isto nos levaria a um sistema de bases de dados múltiplas. Vários SGBD usam uma arquitetura cliente-servidor.

Outros critérios de classificação menos importantes são o custo e o tipo de propósito a que se destina. BD de propósito geral utilizam os sistemas OLAP (on-line Analytical Process) enquanto que os BD de propósito especial utilizam sistemas OLTP (on-line transaction processing) que suportam um grande número de transações concorrentes com pouca perda de performance.

Arquitetura ANSI/SPARC para banco de dados

Algumas características importantes dos sistemas de bancos de dados os diferenciam dos sistemas de arquivos tradicionais. Considere, por exemplo, as características mencionadas em [Elmasri e Navathe, 1994]:

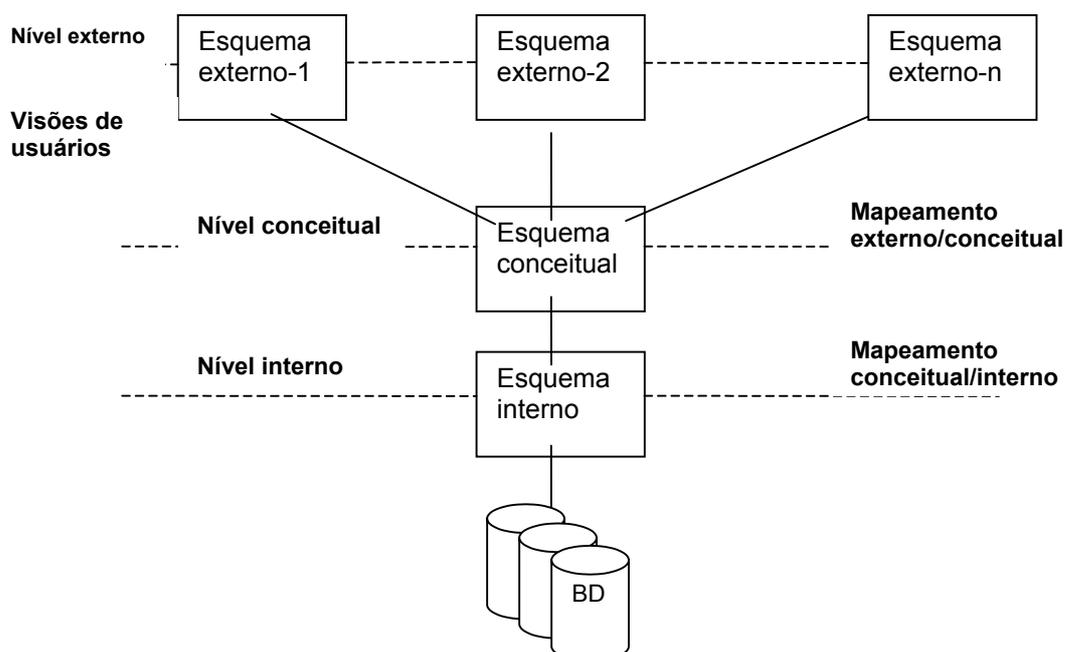
- separação entre programas e dados;
- o suporte para múltiplas visões de usuário;
- o compartilhamento de dados e processamento multiusuário de transações;
- o armazenamento, no banco de dados, de sua própria descrição ou esquema.

Para suportar essas características, um comitê de padronização do ANSI (*American National Standards Institute*) propôs, em 1978, uma arquitetura para sistemas de banco de dados que ficou conhecida como arquitetura ANSI/SPARC ou arquitetura de três esquemas. O objetivo desta arquitetura é separar o banco de dados físico das aplicações do usuário, por meio de três diferentes níveis de esquemas:

- **Esquema interno**, que descreve a estrutura física de armazenamento do banco de dados, a sua organização de arquivos e os seus métodos de acesso.
- **Esquema conceitual**, que 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 de entidades, atributos, relacionamentos, operações do usuário e restrições sobre dados.
- **Esquemas externos**, também chamados visões de usuário, que descrevem as partes do banco de dados que são do interesse de um grupo de usuários, escondendo as demais.

Os esquemas são meramente descrições de dados: o único nível que existe realmente, isto é, que armazena dados, é o nível interno. Como cada grupo de usuários só conhece o seu esquema externo, o SGBD deve transformar cada solicitação de usuário do respectivo para o esquema conceitual e deste para o esquema interno. Tais transformações são realizadas por processos de mapeamento que, em geral, consomem significativo tempo de processamento, às vezes proibitivo, dependendo da aplicação.

A figura 1, a seguir, ilustra a arquitetura de três esquemas.



O conceito de independência de dados pode ser bem entendido nesta arquitetura. A *independência lógica* de dados é a flexibilidade para alterar o esquema conceitual sem a necessidade de alteração dos esquemas externos ou dos programas de aplicação. A *independência física* de dados é a flexibilidade para alterar o esquema interno sem a necessidade de alterar o esquema conceitual ou os esquemas externos.

Mapeamentos

Observando a figura acima, destacam-se dois níveis de mapeamento na arquitetura. O mapeamento conceitual/interno que define a correspondência entre a visão conceitual e o banco de dados armazenado; especifica como os registros e campos conceituais são representados no nível interno. Se a estrutura do banco de dados armazenado for modificada - isto é, se for executada uma mudança na definição da estrutura armazenada – o mapeamento conceitual/interno também deverá ser modificado de acordo, de forma que o esquema conceitual permaneça invariável. (O controle destas mudanças é de responsabilidade do DBA) Em outras palavras, os efeitos dessas modificações devem ser isolados abaixo do nível conceitual, de maneira a preservar a independência de dados.

Um mapeamento externo/conceitual define a correspondência entre uma determinada visão externa e a visão conceitual. As diferenças que podem existir entre estes dois níveis são similares àquelas que podem existir entre a visão conceitual e o banco de dados armazenado. Como exemplo, os campos podem ter tipos de dados diferentes, as denominações de campo e registro podem ser modificadas, campos conceituais múltiplos podem ser combinados num único campo externo (virtual) etc.

Sistemas de Informação apoiados em banco de dados

Um Sistema de Informação apoiado em banco de dados (SIBD) pode ser definido como um conjunto de atividades que regulam o compartilhamento e distribuição de informações e o armazenamento e recuperação de dados que são relevantes ao gerenciamento da organização.

Um SIBD envolve quatro componentes principais: usuários, hardware, software (software aplicativo + SGBD + sistema operacional) e dados (banco de dados).

Principais vantagens de um SIBD

- Rapidez na manipulação e no acesso à informação;
- Redução do esforço humano no desenvolvimento e utilização;
- Disponibilização da informação no tempo necessário;
- Controle integrado de informações distribuídas fisicamente;
- Redução de redundância e de inconsistência de informações;
- Compartilhamento de dados;
- Aplicação automática de restrições de segurança;
- Redução de problemas de integridade.

Principais dificuldades dos SIBD

- *Controle de Concorrência*: a integração de dados, um dos pressupostos básicos da filosofia de banco de dados, tem como contrapartida a necessidade de compartilhamento dos mesmos. Tal compartilhamento requer a adoção de controles que não permitam que aplicações concorrentes levem o banco de dados a um estado de inconsistência;
- *Controle de Integridade*: decorre do fato de que aplicações diferentes podem levar o banco de dados a uma situação de inconsistência requerendo, portanto, um mecanismo que seja independente dessas aplicações e que garanta a sua integridade;

- *Controle de Segurança*: uma vez que os dados foram integrados e necessitam ser compartilhados, tornam-se necessários mecanismos que restrinjam o acesso de usuários a dados para os quais não tenham recebido a devida autorização de acesso;
- *Controle de Recuperação de Falhas*: uma falha num SI convencional afeta apenas aquele sistema. Num SIBD, normalmente integrando diversos SI convencionais, uma falha afetará toda a organização e por esse motivo os SGBD devem ser dotados de mecanismos de proteção contra falhas.

Definição de Sistema Gerenciador de Banco de Dados (SGBD)

É um software com recursos específicos para facilitar a manipulação das informações dos bancos de dados e o desenvolvimento de programas aplicativos. O SGBD também pode ser considerado como um módulo de programa que fornece a interface entre os dados de baixo nível armazenados num banco de dados e os programas aplicativos ou as solicitações submetidas ao sistema. Já C. J. Date define o SGBD como um software que manipula todos os acessos ao banco de dados; proporciona a interface de usuário ao sistema de banco de dados.

Esquemas e Instâncias

Em qualquer modelo de dados utilizado, é importante distinguir a “descrição” do banco de dados do “banco de dados” por si próprio. A descrição de um banco de dados é chamada de *esquema* de um banco de dados e é especificada durante o projeto do banco de dados.

Os dados armazenados em um banco de dados em um determinado instante do tempo formam um conjunto chamado de *instância* do banco de dados. A instância altera toda vez que uma alteração no banco de dados é feita.

O SGBD é responsável por garantir que toda a instância do banco de dados satisfaça ao esquema do banco de dados, respeitando sua estrutura e suas restrições.

Arquitetura Três Esquemas

A principal meta da arquitetura “três esquemas” (figura 2) é separar as aplicações do usuário do banco de dados físico. Os esquemas podem ser definidos como:

- *nível interno*: ou esquema interno, o qual descreve a estrutura de armazenamento físico do banco de dados; utiliza um modelo de dados e descreve detalhadamente os dados armazenados e os caminhos de acesso ao banco de dados;
- *nível conceitual*: ou esquema conceitual, o qual descreve a estrutura do banco de dados como um todo; é uma descrição global do banco de dados, que não fornece detalhes do modo como os dados estão fisicamente armazenados;
- *nível externo*: ou esquema de visão, o qual descreve as visões do banco de dados para um grupo de usuários; cada visão descreve quais porções do banco de dados um grupo de usuários terá acesso.

Independência de Dados

A “independência de dados” pode ser definida como a capacidade de se alterar um esquema em um nível em um banco de dados sem ter que alterar um nível superior (figura 2). Existem dois tipos de independência de dados:

- independência de dados lógica: é a capacidade de alterar o esquema conceitual sem ter que alterar o esquema externo ou as aplicações do usuário;
- independência de dados física: é a capacidade de alterar o esquema interno sem ter que alterar o esquema conceitual, o esquema externo ou as aplicações do usuário.

As Linguagens para Manipulação de Dados

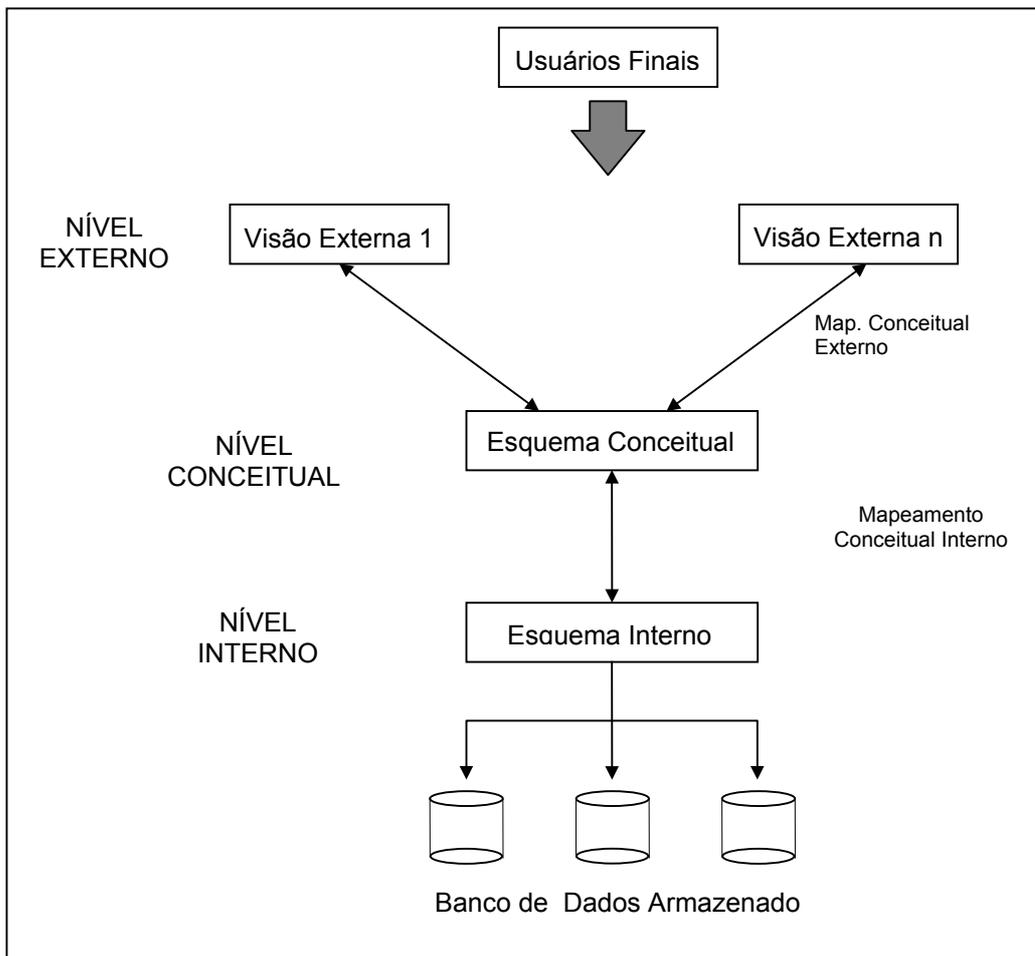
Para a definição dos esquemas conceitual e interno pode-se utilizar uma linguagem chamada DDL (Data Definition Language - Linguagem de Definição de Dados). O SGBD possui um compilador DDL que permite a execução das declarações para identificar as descrições dos esquemas e para armazená-las no catálogo do SGBD. A DDL é utilizada em SGBDs onde a separação entre os níveis interno e conceitual não é muito clara.

Em um SGBD em que a separação entre os níveis conceitual e interno são bem claras, é utilizado uma outra linguagem, a SDL (Storage Definition Language - Linguagem de Definição de Armazenamento) para a especificação do esquema interno. A especificação do esquema conceitual fica por conta da DDL.

Em um SGBD que utiliza a arquitetura três esquemas, é necessária a utilização de mais uma linguagem para a definição de visões, a VDL (Vision Definition Language - Linguagem de Definição de Visões).

Uma vez que o esquema esteja compilado e o banco de dados esteja populado, usa-se uma linguagem para fazer a manipulação dos dados, a DML (Data Manipulation Language - Linguagem de Manipulação de Dados).

Figura 2: Arquitetura ANSI/SPARC



Um SGBD é um sistema complexo, formado por um conjunto muito grande de módulos. A figura 3 mostra um esquema da estrutura de funcionamento de um SGBD.

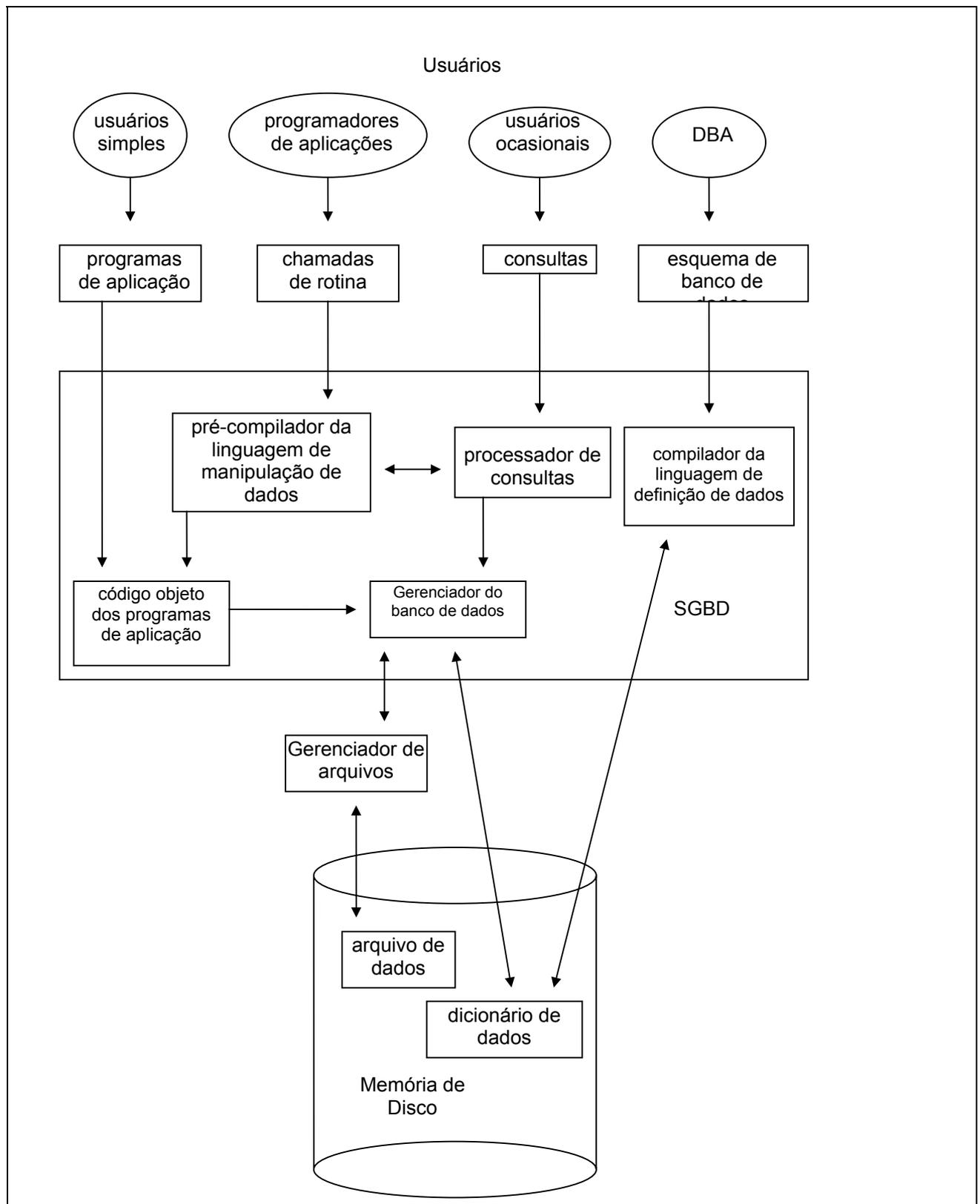


Figura 3: Estrutura de um Sistema Gerenciador de Banco de Dados

2. Modelo de Dados

Fundamentos de modelos de dados

Modelos de dados são utilizados para construir esquemas que são as representações da realidade. A qualidade dos esquemas resultantes não depende somente da habilidade dos projetistas de banco de dados, mas também da qualidade do modelo de dados selecionado.

A pedra fundamental de todos os modelos de dados é uma pequena coleção de mecanismos primitivos de abstração: classificação, agregação e generalização. As abstrações ajudam ao projetista a entender, classificar e modelar a realidade. Através das abstrações, o projetista pode classificar objetos do mundo real e modelar os relacionamentos entre eles.

Abstrações

Uma abstração é um processo mental que usamos quando selecionamos algumas características e propriedades de um conjunto de objetos e excluimos outras características não tão relevantes. Em outras palavras, aplicamos uma abstração sempre que nos concentramos em propriedades de um conjunto de objetos que consideramos essenciais, e esquecemos suas diferenças. Três tipos de abstração são usadas na modelagem conceitual: classificação, agregação e generalização.

Um modelo de dados é, então, uma coleção de conceitos que podem ser usados para descrever um conjunto de dados e operações para manipulação dos dados. Quando um modelo de dados descreve um conjunto de conceitos de uma dada realidade, é denominado Modelo de Dados Conceitual. Os conceitos em um modelo de dados são construídos com o uso de mecanismos de abstração e são descritos através de representações lingüísticas e gráficas, isto é, uma sintaxe pode ser definida e uma notação gráfica pode ser desenvolvida como partes de um modelo de dados.

O Modelo Relacional

O **modelo relacional** foi criado por Codd em 1970 e tem por finalidade representar os dados como uma coleção de relações, onde cada relação é representada por uma **tabela**, ou falando de uma forma mais direta, um arquivo. Porém, um arquivo é mais restrito que uma tabela. Toda tabela pode ser considerada um arquivo, porém, nem todo arquivo pode ser considerado uma tabela.

Quando uma relação é pensada como uma tabela de valores, cada linha nesta tabela representa uma coleção de dados relacionados. Estes valores podem ser interpretados como fatos descrevendo uma instância de uma entidade ou de um relacionamento. O nome da tabela e das colunas desta tabela são utilizados para facilitar a interpretação dos valores armazenados em cada linha da tabela. Todos os valores em uma coluna são necessariamente do mesmo tipo.

Na terminologia do modelo relacional, cada tabela é chamada de **relação**; uma linha de uma tabela é chamada de **tupla**; o nome de cada coluna é chamado de **atributo**; o tipo de dado que descreve cada coluna é chamado de **domínio**.

Domínios, Tuplas, Atributos e Relações

Um **domínio D** é um conjunto de valores atômicos, sendo que por atômico, podemos compreender que cada valor do domínio é indivisível. Durante a especificação do domínio é importante destacar o tipo, o tamanho e a faixa do atributo que está sendo especificado. Por exemplo:

Coluna	Tipo	Tamanho	Faixa
RG	Numérico	10,0	03000000-25999999
Nome	Caracter	30	a-z, A-Z
Salário	Numérico	5,2	00100,00-12999,99

Atributo Chave de uma Relação

Uma relação pode ser definida como um conjunto de tuplas distintas. Isto implica que a combinação dos valores dos atributos em uma tupla não pode se repetir na mesma tabela. Existirá sempre um subconjunto de atributos em uma tabela que garantem que não haverá valores repetidos para as diversas tuplas da mesma, garantindo que $t1[SC] \neq t2[SC]$.

SC é chamada de **superchave** de uma relação. Toda relação possui ao menos uma superchave - o conjunto de todos os seus atributos. Uma **chave C** de uma relação **R** é uma superchave de **R** com a propriedade adicional que removendo qualquer atributo **A** de **K**, resta ainda um conjunto de atributos **K'** que não é uma superchave de **R**. Uma chave é uma superchave da qual não se pode extrair atributos. Por exemplo, o conjunto: *(RA, Nome, Endereço)* é uma superchave para estudante, porém, não é uma chave pois se tirarmos o campo *Endereço* continuaremos a ter uma superchave. Já o conjunto *(Nome da Revista, Volume, N^o da Revista)* é uma superchave e uma chave, pois qualquer um dos atributos que retirarmos, deixaremos de ter uma superchave, ou seja, *(Nome da Revista, Volume)* não identifica uma única tupla.

Em outras palavras, uma superchave é uma **chave composta**, ou seja, uma chave formada por mais que um atributo. Veja o exemplo abaixo:



Tabela DEPENDENTES				
<u>RG Responsável</u>	<u>Nome Dependente</u>	Dt. Nascimento	Relação	Sexo
10101010	Jorge	27/12/86	Filho	Masculino
10101010	Luiz	18/11/79	Filho	Masculino
20202020	Fernanda	14/02/69	Conjuge	Feminino
20202020	Angelo	10/02/95	Filho	Masculino
30303030	Fernanda	01/05/90	Filho	Feminino

Quando uma relação possui mais que uma chave (não confundir com chave composta) - como por exemplo RG e CIC para empregados - cada uma destas chaves são chamadas de **chaves candidatas**. Uma destas chaves candidatas deve ser escolhida como **chave primária**.

Uma **chave estrangeira CE** de uma tabela **R₁** em **R₂** ou vice-versa, especifica um relacionamento entre as tabelas **R₁** e **R₂**.

Mapeamento do Modelo Entidade Relacionamento para o Modelo Relacional

O mapeamento do modelo entidade relacionamento para o Modelo Relacional segue oito passos básicos a saber:

1. Para cada entidade **E** no modelo ER é criada uma tabela **T₁** no Modelo Relacional que inclua todos os atributos simples de **E**; para cada atributo composto, são inseridos apenas os componentes simples de cada um; um dos atributos chaves de **E** deve ser escolhida como a chave primária de **T₁**;
2. Para cada entidade fraca **EF** com entidade proprietária **E** no modelo ER, é criada uma tabela **T₁** no Modelo Relacional incluindo todos os atributos simples de **EF**; para cada atributo composto, são inseridos apenas os componentes simples de cada um; a chave primária desta relação **T₁** será composta pela chave parcial da entidade fraca **EF** mais a chave primária da entidade proprietária **E**;

3. Para cada relacionamento regular com cardinalidade 1:1 entre entidades E_1 e E_2 que geraram as tabelas T_1 e T_2 respectivamente, devemos escolher a chave primária de uma das relações (T_1 , T_2) e inseri-la como chave estrangeira na outra relação; se um dos lados do relacionamento tiver participação total e outro parcial, então é interessante que a chave do lado com participação **parcial** seja inserido como chave estrangeira no lado que tem participação **total**;
4. Para cada relacionamento regular com cardinalidade 1:N entre entidades E_1 e E_2 respectivamente e que geraram as tabelas T_1 e T_2 respectivamente, deve-se inserir a chave primária de T_1 como chave estrangeira em T_2 ;
5. Para cada relacionamento regular com cardinalidade N:N entre entidades E_1 e E_2 , cria-se uma nova tabela T_1 , contendo todos os atributos do relacionamento mais o atributo chave de E_1 e o atributo chave de E_2 ; a chave primária de T_1 será composta pelos atributos chave de E_1 e E_2 ;
6. Para cada atributo multivalorado A_1 , cria-se uma tabela T_1 , contendo o atributo multivalorado A_1 , mais o atributo chave C da tabela que representa a entidade ou relacionamento que contém A_1 ; a chave primária de T_1 será composta por A_1 mais C ; se A_1 for composto, então a tabela T_1 deverá conter todos os atributos de A_1 ;
7. Para cada relacionamento n-ário, $n > 2$, cria-se uma tabela T_1 , contendo todos os atributos do relacionamento; a chave primária de T_1 será composta pelos atributos chaves das entidades participantes do relacionamento;
8. Converta cada especialização com m subclasses $\{S_1, S_2, \dots, S_m\}$ e superclasse SC , onde os atributos de SC são $\{c, a_1, a_2, \dots, a_n\}$ onde c é a chave primária de SC , em tabelas utilizando uma das seguintes opções:
 - 8.1. Crie uma tabela T para SC com os atributos $A(T) = \{c, a_1, a_2, \dots, a_n\}$ e chave $C(T) = c$; crie uma tabela T_i para cada subclasse S_i , $1 \leq i \leq m$, com os atributos $A(T_i) = \{c\} \cup A(S_i)$, onde $C(T) = c$;
 - 8.2. Crie uma tabela T_i para cada subclasse S_i , $1 \leq i \leq m$, com os atributos $A(T_i) = A(S_i) \cup \{c, a_1, a_2, \dots, a_n\}$ e $C(T_i) = c$;
 - 8.3. Crie uma tabela T com os atributos $A(T) = \{c, a_1, a_2, \dots, a_n\} \cup A(S_1) \cup \dots \cup A(S_m) \cup \{t\}$ e $C(T) = c$, onde t é um atributo **tipo** que indica a subclasse à qual cada tupla pertence, caso isto venha a ocorrer;
 - 8.4. Crie uma tabela T com atributos $A(T) = \{c, a_1, a_2, \dots, a_n\} \cup A(S_1) \cup \dots \cup A(S_m) \cup \{t_1, t_2, \dots, t_m\}$ e $C(T) = c$; esta opção é para generalizações com “overlapping”, e cada t_i , $1 \leq i \leq m$, é um atributo “booleano” indicando se a tupla pertence ou não à subclasse S_i ; embora funcional, esta opção pode gerar uma quantidade muito grande de valores nulos;

Modelos de dados semânticos (modelo entidade-relacionamento)

Qual seria o problema?

A maioria dos sistemas de bancos de dados tem um conhecimento muito limitado sobre o significado dos seus dados. Eles conseguem “compreender” certos valores de dados simples e certos relacionamentos. Tal incompreensão não permite que os sistemas respondam de maneira mais inteligente as interações dos usuários. Por exemplo, os sistemas de banco de dados não conseguem entender que os pesos de peças e as quantidades de expedição, embora representem valores numéricos, são diferentes em tipo – i.e., semanticamente diferentes.

O termo modelo semântico de dados, algumas vezes utilizado para um ou outro dos modelos “ampliados”, não é particularmente adequado. De outro lado, a “modelagem semântica” é uma classificação apropriada para a atividade global de tentativas de representar o sentido.

A Abordagem Global

Podemos caracterizar a abordagem global do problema da modelagem semântica como segue:

1. Primeiro vamos padronizar um conjunto de conceitos semânticos úteis. Como exemplo vamos citar *entidade, propriedade e associação*. Informalmente podemos concordar que o mundo real compõe-se de entidades que possuem propriedades, são conectadas em associações, etc.
2. A seguir, determinamos um conjunto de objetos simbólicos correspondentes para representar os conceitos semânticos.
3. Determinamos, também, um conjunto de regras de integridade ao lado dos objetos simbólicos.
4. Por fim, desenvolvemos um conjunto de operadores para a manipulação dos objetos simbólicos.

Alguns Conceitos Semânticos úteis

CONCEITO	Definição Formal	Exemplos
ENTIDADE	Um objeto passível de distinção (de um tipo particular)	Fornecedor, Peça, Expedição, Funcionário, Departamento, Pessoa, Orquestra, Ordem de Compra
PROPRIEDADE	Uma peça de informação que descreve uma entidade	Código do Fornecedor, Quantidade de Expedição, Departamento do Funcionário, Altura da Pessoa
ASSOCIAÇÃO	Um relacionamento de muitos-para-muitos entre Entidades	Expedição (Peça do Fornecedor), Atribuição (departamento do funcionário)
SUBTIPO	O tipo de entidade Y é um subtipo da entidade X se, e somente se, cada Y for necessariamente um X	Funcionário é um subtipo de Pessoa, Concerto é um subtipo de composição.

Uma das principais propostas da área da modelagem semântica foi o “modelo entidade/relacionamento proposto por Peter Chen em 1976. Com o próprio modelo, Chen introduziu uma técnica de diagramação (“diagramas de entidade/relacionamento”) para representação da estrutura lógica do banco de dados. A popularidade da “modelagem de entidade/relacionamento” como abordagem de design do banco de dados provavelmente se deve mais a existência da técnica de diagramação do que a qualquer outra causa.

Exercícios

Para cada Mini-Mundo apresentado a seguir faça o seu respectivo Modelo Conceitual de Dados, contendo:

- a) Diagrama ERA completo (Entidades, Relacionamentos, Atributos simples e identificador, Cardinalidades e Repetições)
- b) Dicionários de Dados (nome e descrição das Entidades, Relacionamentos e Atributos)
- c) Justificativa das cardinalidades, envolvendo a “leitura” e o “porque” (do mínimo e do máximo)
- d) Justificativa das Repetições 1, envolvendo a “leitura” e o “porque”

Mini-Mundo 1: (discutir: cardinalidade)

A administradora de Imóveis "IMOR Tal" é uma empresa que cuida, principalmente, da compra e venda de imóveis residenciais e comerciais no Grande Rio, dentre outras atividades. O atendimento atualmente é demorado e, muitas vezes, incompleto devido a demora no manuseio de muitas fichas, acarretando a perda de muitas oportunidades de negócio. Todos os imóveis são comprados pela imobiliária para, então, serem colocados a venda. A direção da empresa definiu como prioridade automatizar o processo de comercialização (compra e venda) dos imóveis, envolvendo seus proprietários (novos e antigos). A imobiliária considera "proprietário" toda pessoa que participou de um processo de comercialização (compra ou venda) no papel de dono (antigo ou novo). Entre outras informações, o sistema deverá ser capaz de controlar os imóveis comprados, vendidos e os de seu "interesse" (não foram comercializados), e emitir:

- a) Relação de todos os imóveis disponíveis para venda, contendo para cada um: Endereço, Bairro, Área (m²), descrição, Proprietário antigo (o atual é a administradora) e o Preço Mínimo para venda
- b) Relação de todos os imóveis vendidos, por bairro, contendo para cada um: Bairro, Proprietário antigo, Proprietário novo, Preço de venda (ao proprietário novo) e o Preço de compra (pela imobiliária)
- c) Relação dos proprietários que compraram mais de um imóvel na imobiliária (nome, CPF, endereço, telefone)
- d) Relação dos proprietários que venderam mais de um imóvel para a imobiliária (nome, telefone)

Mini-Mundo 2-A: (discutir: modelo de conjunto)

Uma loja de venda de Eletrodomésticos quer automatizar o seu controle de compra e troca de aparelhos por parte de seus clientes. Todo aparelho vendido possui garantia de 1 ano, a partir da data de venda. Isto significa que qualquer troca só poderá ser realizada dentro deste período, mesmo que já tenha havido várias trocas em função desta compra.

No termo de garantia é anotado a data da compra, marca, modelo e número de série do aparelho vendido juntamente com o nome e endereço do cliente que o comprou. A cada troca de aparelho, relativo a primeira compra, é verificado se ainda está no prazo de garantia, e é registrado o cliente que realizou a troca. Qualquer cliente pode realizar uma troca, mesmo que não tenha sido o comprador. Os aparelhos defeituosos são devolvidos para a fábrica e não mais retornam para a loja. A loja quer saber:

- a) Relação de aparelhos disponíveis na loja
- b) Relação de aparelhos que apresentaram defeitos contendo quem realizou a troca, a data e o defeito apresentado
- c) Relação de clientes cujas compras nunca apresentaram defeito

Mini-Mundo 2-B:(discutir: modelo de conjunto)

Suponha que no mini-mundo anterior, a loja decida que somente o próprio comprador é quem pode realizar a troca. Quais as implicações disto? Gere um novo modelo com este enfoque.

Mini-Mundo 3: (discutir: habilitação e tipo)

Uma fábrica de roupas exclusivas (cada modelo, único, e projetado por estilistas famosos) deseja um sistema para controlar sua produção. A fábrica conta atualmente com 1230 funcionários sendo que a maior parte dos mesmos são costureiras trabalhando na atividade fim. A fábrica possui aproximadamente 600 máquinas de costura de diversos tipos (overlock, zig-zag, costura reta, etc.) de diversos fabricantes. Para ingressar como costureira, a funcionária é avaliada para determinar em

que tipo de máquina ela possui habilitação. Cada máquina pode realizar um ou mais tipos de costura.

Cada peça de roupa é produzida integralmente por uma costureira em uma máquina, sendo que neste período nem a costureira, nem a máquina podem ser alocados para outra coisa.

A remuneração das costureiras é mensal baseado em uma alíquota fixa (15%) sobre o preço de venda de cada peça. Nenhuma costureira pode receber menos que um determinado valor mínimo que é negociado no momento da contratação de cada uma. As costureiras são divididas em supervisões, cada uma possuindo uma supervisora que é a responsável pela qualidade do que é produzido, e pela monitoração das máquinas que estão em conserto. A máquina só vai para conserto após o término da produção da peça. A supervisora da costureira que estava produzindo nesta máquina se torna a responsável pela monitoração de seu conserto.

A fábrica necessita das seguintes informações:

a) Relatório de peças produzidas por uma costureira num determinado período, no seguinte formato: modelo da peça, descrição do modelo, data e hora de início e término da fabricação, código da máquina de costura, localização da máquina e o fabricante.

b) Relatório das costureiras sem produção no período (matrícula da costureira, nome, Valor Mínimo Negociado).

c) Relação das máquinas disponíveis, informando para cada uma o seu fabricante e o(s) tipo(s) de costura que possui.

d) Quais costureiras estão disponíveis e habilitadas a trabalhar em um tipo de máquina no momento?

e) Relação das máquinas que estiveram mais de 10 vezes em conserto, contendo: código da máquina e para cada conserto, matrícula e nome da supervisora responsável, data início e término do conserto.

Mini-Mundo 6:

Uma firma que utiliza equipamentos de informática necessita de um sistema que gerencie a sua rede de microcomputadores (ponto-a-ponto, não existe servidor da rede), controlando usuários, máquinas e impressoras. A rede é composta de servidores de impressão, estações e impressoras. O sistema também irá controlar a partir de qual estação o usuário está conectado a rede, e os seus arquivos enviados para impressão.

Para todos os micros deseja-se cadastrar: código do patrimônio, descrição, capacidade do disco rígido, quantidade de memória e, sendo uma estação a sua localização, sendo um servidor o tamanho máximo do buffer e a quantidade máxima de buffers de impressão e que ele suporta, e as impressoras ligadas a ele (no máximo 3), caso existam. Para impressoras deseja-se cadastrar: código do patrimônio, descrição, velocidade (CPS) e, conseqüentemente, o servidor a que está ligada. Nesta firma todas as impressoras estão ligadas a algum servidor, não sendo compartilhada por mais de um servidor.

Para controlar os usuários, o sistema só precisa do nome de guerra e senha de cada um. Como os usuários não possuem máquina fixa, a sua conexão à rede pode ocorrer a partir de qualquer estação. Tendo o usuário uma conexão ativa, o sistema não permitirá que ele se conecte a partir de outra estação. Há interesse em controlar apenas as conexões ativas (as conexões desfeitas são irrelevantes).

No caso de impressão, o sistema deverá saber qual o arquivo, de quem ele é, e em qual impressora será impresso (atenção: somente usuários com conexão ativa e que possuem condição de enviar arquivos para impressão). É o usuário que escolhe a impressora onde ele quer que o seu arquivo seja impresso. Nada impede que usuários diferentes enviem arquivos de mesmo nome para impressão, porém (nesta firma) para o mesmo usuário isso não é possível, mesmo em impressoras diferentes. Neste caso o sistema permite alterar o número de cópias a serem impressas. Só deve ser mantido registro dos arquivos que ainda estão na fila de impressão.

O sistema deverá listar, para cada impressora, os arquivos que estão aguardando impressão, com o respectivo usuário que a enviou, mesmo que o usuário não esteja mais ativo na rede. Sempre que solicitado o sistema exibirá, para cada estação, o seu código e, caso exista, o nome do usuário conectado, a data e hora início desta conexão e, se houver, nome e quantidade de cópias dos arquivos que ele enviou e que ainda estão aguardando impressão.

Mini-Mundo 7-A: (discutir: generalização - especialização)

Um Vídeo Clube deseja controlar o empréstimo de fitas a seus sócios. Atualmente o controle de empréstimo é feito utilizando dois conjuntos de fichas: o das informações dos sócios e o das informações das cópias, no qual se controla também os empréstimos. Nas fichas dos sócios constam as seguintes informações: número de inscrição, nome, endereço e telefone. Nas fichas das cópias estão registrados: código de identificação da fita, título do filme, duração, ano, gênero, nome do diretor e de 2 (dois) dos artistas principais, data de aquisição e o estado da fita (bom ou ruim), além de uma lista de empréstimos com número de inscrição do sócio, data do empréstimo, data da devolução e valor pago. Estes dois últimos são preenchidos quando a cópia é devolvida.

Nesta locadora existem vários filmes com várias cópias, e cada cópia recebe uma etiqueta com um código, por exemplo: 5.315 (como se fosse o número do CONCINE, único para cada cópia original).

Um empréstimo é válido por 72 horas no máximo e possui preço fixo. A locadora pretende colocar terminais de consulta em sua loja, possibilitando seus sócios a escolha de filmes a partir do gênero, diretor ou dos artistas prediletos. Não é aceita reserva e todos os pagamentos são feitos no momento da devolução. Se o sócio atrasar a devolução deverá pagar multa.

A gerência deseja que o sistema forneça um relatório de todos os filmes que possui, informando a quantidade de cópias por filme, além de relatórios de filmes por gênero, diretor e artista. Ela deseja também uma lista das cópias em mal estado e outra de sócios inadimplentes. Existe interesse em manter um controle sobre diretores e artistas com nome, país de origem e data de nascimento de cada um.

Mini-Mundo 7-B: (discutir: entidade)

Troque o segundo parágrafo do mini-mundo 7-A, pelo abaixo:

Nesta locadora existem vários filmes com várias cópias, e cada cópia recebe uma etiqueta com um código, por exemplo: P305-03, que significa Policial (P), Filme número 305, 3ª cópia deste filme.

Mini-Mundo 9: (discutir: cardinalidade)

Um dos restaurantes mais tradicionais do Rio de Janeiro teve sempre o seu controle realizado manualmente pelos seus proprietários nipônicos: Joaquim, Manoel e João. A sorte "lotérica" bateu à porta de Manoel, que retornou imediatamente para sua terra natal. Com o crescimento do movimento financeiro do restaurante, João e Joaquim passaram a se desentender quanto a melhor maneira de administrar o restaurante, resultando na compra da parte de João por Joaquim. Desta maneira, Joaquim assumiu a posse integral do restaurante.

O restaurante possui atualmente 30 garçons, servindo diariamente mais de 500 refeições e o cardápio oferece mais de 40 pratos diferentes. O restaurante conta com uma ampla área de mais de 1000 m² e dispõe de mesas de 2, 4 e 6 lugares num total geral de 80 mesas. O horário de funcionamento atual é de 11:00h às 2:00h. Cada garçom é responsável por atender no mínimo 4 mesas e no máximo 10, não podendo atender nenhuma mesa fora de sua responsabilidade. A remuneração é um percentual fixo sobre o consumo das mesas que cada um atendeu.

Ao encerrar a conta, o cliente preenche uma avaliação sobre o atendimento prestado pelo garçom. Ao final do mês, os garçons que obtiverem as 3 melhores médias de desempenho recebem uma gratificação extra. O restaurante impõe que não pode existir mais de um garçom atendendo ao



mesmo cliente na mesa, e que este garçom deverá realizar todo o atendimento até o cliente ir embora, mesmo que ultrapasse seu horário normal de trabalho. Não é permitido que pessoas ocupem mesas sem consumir.

Periodicamente Joaquim necessita das seguintes informações:

- a) Relação das mesas que um garçom tem sob sua responsabilidade.
- b) Número de assentos que um garçom tem sob sua responsabilidade.
- c) Lista dos pratos servidos em uma mesa durante um certo período de tempo.
- d) Salário a pagar a cada garçom no final do mês.
- e) Lista dos pratos mais consumidos por dia da semana.
- f) Relação das mesas que por mais tempo estiveram ocupadas.
- g) Relação dos garçons que devem receber gratificação ao final do mês junto com suas respectivas médias.
- h) Qual(is) garçom(ns) é(são) responsável(eis) por uma determinada mesa.

Mini-Mundo 10-A: (discutir: triplo/formas de agregação.)

A empresa Soft Lie deseja automatizar o controle gerencial do seu Departamento de Recrutamento e Seleção. Qualquer pessoa, para se inscrever como candidato, tem que ser indicado por um funcionário para um cargo específico. Todo candidato possui um número de inscrição único, mesmo que tenha recebido varias indicações, ou se candidate a vários cargos (um funcionário não pode indicar o mesmo candidato mais de uma vez para o mesmo cargo).

Cada cargo possui testes específicos que deverão ser realizados pelos candidatos, como parte do processo seletivo. Um candidato só é considerado apto para a fase de entrevista se tiver nota superior a 8 em todos os testes previstos. No caso do candidato se inscrever um mais de um cargo que exija o mesmo teste, este deverá ser realizado apenas uma única vez. Durante os seis meses de validade de cada teste, este não poderá ser repetido pelo candidato, devendo ser utilizado a nota tirada (após os seis meses os testes são destruídos). Os testes são corrigidos por funcionários que ocupam um cargo para o qual o teste se aplica, exceto os testes de candidatos indicados por eles.

A Gerência necessita das seguintes informações (dentre outras):

- a) Relação dos candidatos (número de inscrição, CPF, nome) e seus respectivos testes já realizados (código do teste, data) mas ainda não corrigidos, e os cargos (descrição) para os quais cada um foi indicado.
- b) Para um determinado candidato, informe o CPF, nome, matrícula, cargo (código) e o departamento (código e nome) de todos os funcionários que indicaram esse candidato (não importando o cargo para o qual ele foi indicado).
- c) Relação dos funcionários oriundos do processo seletivo normal (ou seja, foram candidatos), com seus testes realizados e as notas.
- d) Para um determinado candidato, informe o código e nome de todos os cargos para o qual ele se candidatou.
- e) Informe todos os candidatos que foram indicados por um determinado funcionário.
- f) Informe todos os candidatos que se inscreveram para um determinado cargo.

Mini-Mundo 10-B:

Troque o primeiro parágrafo do mini-mundo 10-A, pelo abaixo:

A empresa Soft Lie deseja automatizar o controle gerencial do seu Departamento de Recrutamento e Seleção. Qualquer pessoa pode se inscrever como candidato para um ou mais cargos específicos, e recebe um número de inscrição para cada cargo em que ele se inscreva. Só se aceitam inscrições de candidatos para um determinado cargo se ele tiver pelo menos uma indicação de um funcionário que esteja ocupando aquele cargo.

Mini-Mundo 11-A:

Uma loja de departamentos deseja controlar a entrada e saída de produtos dos seus vários

almoxarifados. Cada entrada é endereçada a um almoxarifado e composta de um e somente um produto. Possui uma data de entrada do produto no almoxarifado e uma quantidade. Em cada almoxarifado existe no máximo uma entrada de cada produto, por dia. A maior parte dos produtos é produzido pelas fábricas que a loja possui sendo deixada uma pequena parcela para fornecedores externos. Não há interesse em se controlar as fábricas pertencentes a loja.

Cada saída é composta de um e somente um produto de um almoxarifado, uma data e uma quantidade. Todas as saídas de produto tem um, e somente um, empregado responsável. Só há interesse em monitorar os fornecedores externos que efetivamente tenham fornecido algum produto à loja.

Há interesse da loja em obter as seguintes informações do sistema:

- a) Quantidade de determinado produto em um determinado almoxarifado.
- b) Entradas de um produto em um almoxarifado em um determinado período informando: código do produto, descrição do produto, número do almoxarifado, localização do almoxarifado, data da entrada, quantidade e se for o caso, código e nome do fornecedor externo.
- c) Saídas de um produto de um almoxarifado em um determinado período informando: código do produto, descrição do produto, número do almoxarifado, localização do almoxarifado, data da saída, quantidade, matrícula, nome e função do empregado responsável.
- d) Relação dos produtos que a loja comercializa ou tem intenção de comercializar.
- e) Relação de todos os almoxarifados da loja (ativos e inativos).

Mini-Mundo 11-B: (discutir: identificação de relacionamento)

Suponha que no mini-mundo 11-A, cada entrada possua um número seqüencial único que permite identificá-la, independente do almoxarifado. Que mudanças ocorreriam no modelo?

Mini-Mundo 11-C:

Suponha que no mini-mundo 11-A, cada entrada possua um número seqüencial que permite identificá-la, em cada almoxarifado. Que mudanças ocorreriam no modelo?

Mini-Mundo 12: (discutir: tabela de faixas e card.)

A polícia rodoviária federal quer automatizar o controle de aplicação de multas. Faça um modelo conceitual a partir das especificações a seguir. Cada infração possui um código, uma descrição e um valor baseado em uma unidade monetária que não sofre os efeitos da inflação, chamada de unidade monetária de trânsito (UMT).

Apesar da polícia utilizar o cadastro de veículos do DETRAN, para a polícia, os veículos automotores são divididos em 3 categorias: automóveis de passeio, ônibus e caminhões. Dependendo da categoria, existe uma fórmula de cálculo para a multa (atenção: multa e o fato de um veículo cometer uma infração definida no código) que funciona da seguinte maneira:

- a) Para os automóveis de passeio é aplicado um fator multiplicador sobre o valor da infração de acordo com a potência do motor. Por exemplo para os carros com potência até 50hp o fator multiplicador é 1. Para aqueles entre 50 e 70 o fator é 1.2 e assim por diante.
- b) Para os ônibus, o fator depende da capacidade de transporte de passageiros. Por exemplo, ônibus que transportam até 20 passageiros aplica-se o fator 1. Para aqueles com capacidade entre 20 e 40 aplica-se o fator 1.1 e assim por diante.
- c) Nos caminhões, o fator varia de acordo com o número de rodas. Caminhões com 4 rodas tem associado o fator 1. Caminhões entre 4 e 8 aplica-se o fator 1.2 e assim por diante.

Cada multa está associada a uma data/hora e local de ocorrência. Não é permitido cadastrar mais de uma multa para um veículo que infringiu uma mesma infração numa mesma data/hora e local. O proprietário pode recorrer das multas recebidas, quantas vezes achar necessário, porém cada recurso só poderá ser impetrado sobre uma única multa, e só será julgado até a terceira instância. Todo recurso possui um número e um motivo e, automaticamente, é aberta uma instância

(a primeira). Toda instância possui um número e, após julgada é colocado a data e o parecer.

Cada veículo pertence a um e somente um proprietário num dado instante, sendo que o sistema não está interessado em monitorar antigos proprietários. Entre as informações que a polícia deseja extrair dos sistema encontram-se:

a) Relação das multas de um veículo com os seguintes dados: data, hora, local de ocorrência, código da infração, descrição da infração, valor a ser pago, ano de fabricação, cor do veículo, identidade, nome e endereço do proprietário. Se o veículo for um automóvel informar também sua potência, marca e modelo; caso seja um ônibus sua capacidade de transporte de passageiros e no caso de ser um caminhão o número de rodas e toneladas máxima.

b) Relação dos recursos que um proprietário impetrou, contendo para cada recurso seu número, descrição, instância atual com o parecer (se houver), placa do veículo multado (mesmo que não lhe pertença mais), infração cometida e o valor da multa.

c) Relação das infrações existentes no sistema com seus respectivos códigos, descrições e valor em UMT.

d) Relação dos veículos existentes com seus respectivos proprietários.

Mini-Mundo 13:

Uma firma que presta serviços de limpeza deseja um sistema automatizado que cuide, principalmente, da alocação dos empregados aos pedidos de serviço e de possíveis críticas necessárias no momento da alocação. Não interessa controlar os orçamentos destes serviços.

Cada pedido de serviço é cadastrado para permitir a posterior crítica na alocação dos empregados que vão executá-lo. Ele não recebe número mas é informada a metragem e a data para a sua realização. A firma mantém uma tabela de funções que são necessárias a cada serviço, com a quantidade de homem por metro quadrado, que é útil para efeito da crítica da alocação de mão-de-obra.

Como cada empregado (matrícula e nome) é habilitado a executar uma única função, antes de alocá-lo deve-se criticar se a função dele é prevista naquele tipo de serviço, assim como, se não está ultrapassando a quantidade por metro quadrado prevista daquela função naquele tipo de serviço. Por outro lado, como o empregado passa o dia todo no serviço, deve-se criticar se ele já não está alocado a outro serviço naquele dia. Um serviço sempre começa e termina no mesmo dia e leva o dia inteiro, todo cliente possui um pedido no mínimo e só pode possuir, no máximo, um pedido para cada tipo de serviço por dia. O sistema deve ser capaz de emitir:

a) "Relatório dos pedidos de serviços com alocação incompleta". Imprime CGC e nome do cliente, descrição do serviço e data prevista para execução de todos os pedidos de serviço que ainda não alocaram a quantidade de empregados prevista.

b) "Tabela de Serviços". Imprime código, descrição e valor por metro quadrado de cada serviço e todas as funções necessárias a sua execução com quantidade de homem por metro quadrado.

c) "Tabela de Funções". Imprime código e descrição da função. Imprime, também, salário base, no caso de ser função de alto nível (encarregado, coordenador, etc.), ou nível de instrução mínimo exigido, no caso de ser função especializada (função técnica).

Mini-Mundo 15-A: (discutir: agregação)

O SUS está interessado em controlar os pacientes internados, e seus atendimentos, nos seus hospitais. Quando uma pessoa credenciada junto ao SUS passa mal, ela se dirige a um dos hospitais e se consulta com algum médico. Dependendo da gravidade o(s) médico(s) pode(m) decidir pela internação. Os pacientes, pessoas credenciadas que foram internadas, podem receber atendimento de vários médicos e enfermeiras durante o período de internação. Não há interesse em controlar as pessoas que não foram internadas, nem as consultas antes da internação.

Cada empregado do SUS (médico ou enfermeira) só pode estar vinculado a no máximo 3

hospitais. Não se admite um empregado com mais de um vínculo no mesmo hospital. Não há interesse em controlar as datas em que ocorreram os atendimentos. Há necessidade de se conseguir as seguintes informações:

- a) Relação dos pacientes (nome, código do seguro social, idade) internados num hospital juntamente com os nomes e números dos médicos responsáveis por cada internação, e o período de internação.
- b) Relação dos médicos e enfermeiras (nome, matrícula) que trabalham determinado hospital.
- c) Relação dos médicos (nome, matrícula, especialidade) e enfermeiras (nome, matrícula, cargo) que deram atendimento a um paciente durante uma internação.
- d) Relação dos hospitais (nome, código e endereço) que um médico ou enfermeira mantém vínculo.

Mini-Mundo 17: (discutir: generalização - especialização) BIAGIOTTI

Uma outra corretora, a corretora "ENGAN Ações" deseja um sistema automatizado que a auxilie durante o pregão a comprar e vender ações para/de seus clientes.

A intenção do cliente em negociar uma ação é expressada pela ordem de compra ou pela ordem de venda. Para uma determinada ação, o cliente poderá emitir várias ordens por dia. Uma ordem vale por todo o dia (e somente naquele dia) e, conforme vai sendo negociada, para cada lote fechado cadastra-se a corretora que negociou juntamente com a quantidade, o preço unitário da ação e o momento (hora) da negociação, com isso, a mesma ordem pode acabar sendo negociada aos poucos por várias corretoras e/ou pela mesma.

Na ordem de compra o cliente informa qual a ação e o valor total que ele deseja gastar para adquiri-la, independente do preço da ação. A função da corretora Boa Ação é tentar fechar a maior quantidade dentro deste valor. Na ordem de venda o cliente informa a ação e a quantidade total que ele quer vender e aí a corretora Boa Ação vai tentar fechar o preço mais alto que encontrar. Neste caso é necessário conferir se o cliente tem disponível as ações que colocou à venda. Para tal, devem ser levadas em consideração as compras e vendas desta ação já realizadas pelo cliente, exceto as compras do mesmo dia pois uma ação comprada só estará disponível para venda no dia seguinte.

O sistema deve ser capaz de:

- a) Listar matrícula do cliente, código e nome da ação, data e hora de todas as ordens cadastradas que não foram negociadas, nem parcialmente. Acrescentar valor total em caso de compra, ou quantidade total em caso de venda.
- b) Listar matrícula do cliente, código da ação, data e hora de todas as ordens cadastradas que foram negociadas parcialmente, juntamente com cgc e nome da corretora, momento, quantidade e preço de cada negociação (lote) dentro desta ordem.

Mini-Mundo 19: (discutir: auto-relacionamento)

Uma cidade resolveu realizar um campeonato de Dupla de Tênis (dupla x dupla) entre alunos de bairros diferentes (existe um cadastro de bairros contendo código, nome e população ativa). Para isso fez-se um levantamento de todas as escolas da cidade (nome, bairro, quantidade de alunos). O nome da escola é único no bairro (podendo haver em bairros diferentes). Também foi realizado um levantamento de todos os alunos capacitados para este torneio (código do atleta, nome do atleta, nome da escola, bairro, idade), porém cada escola só pode inscrever 20 atletas no máximo.

Os jogadores podem se organizar em diversas duplas (porém um jogador só pode participar em uma única dupla), desde que sejam de uma escola do mesmo bairro. O Sistema deve poder cadastrar a data, hora e a escola na qual será realizada cada partida entre duplas (sempre de bairros diferentes). Há também o interesse em saber todos os jogadores de um determinado bairro, todas as partidas que serão realizadas em uma escola e as partidas numa determinada data. Sabe-se que as duplas podem jogar várias vezes entre si, em datas diferentes, e que todas as escolas realizarão no mínimo 1 jogo (partida).

Mini-Mundo 23:

O "Banco GHOST" é um banco virtual que opera em âmbito nacional (sem o conceito de agências), onde os correntistas só podem utilizar seus serviços por meio de acesso telefônico (voz), fax, telex, modem, ou acesso via Internet. Neste Banco cada correntista possui apenas uma única conta corrente, e não existe a co-titularidade sobre a conta corrente. O principal objetivo do sistema é gerenciar o cadastro de correntistas, e as informações relevantes para obtenção de descontos por parte dos correntistas.

No momento do cadastro do correntista é fornecido o número da sua conta corrente, bem como o seu limite de crédito e, no caso de pessoa física, é registrado seu cpf, nome, endereço e telefone (neste momento é consultado sobre todos os seus parentes que possuem conta no banco).

O banco oferece desconto na utilização dos serviços bancários aos correntistas que tiverem vínculo empregatício com uma indústria que seja correntista do banco, e aos correntistas que tiverem algum grau de parentesco com outro correntista (1-irmão, 2-pai/filho, 3-avô/neto, 4-tio/sobrinho, etc.). A empresa é que define o tipo de parentesco que lhe interessa.

As indústrias são cadastradas com seu CGC, endereço, razão social, telefone, quantidade de empregados e uma relação dos 5 principais tipos de produtos que ela pode produzir.

No momento da abertura de conta das empresas cujo o propósito é o comércio, deve ser informado a sua inscrição estadual, seu nome fantasia, endereço e telefone, além de seu CGC, razão social e uma relação com todos os tipos de produtos comercializados,. O banco mantém uma lista apenas com os produtos que são produzidos, ou comercializados por seus correntistas.

O sistema devera ter condições de emitir os seguintes relatórios:

- a) Relação de todas as empresas correntistas do banco, que não são nem indústria, nem comércio.
- b) Relação de todas as empresas correntistas do banco, que são indústria e comércio.
- c) Relação dos correntistas que são empregados em uma determinada indústria que é correntista.
- d) Relação dos correntistas que são parentes de um determinado correntista, com tipo "avô/neto".
- e) Relação das empresas que comercializam algum tipo de produto que produzem.
- f) Relação dos produtos produzidos, mas não comercializados por correntistas, contendo o código e a descrição de cada produto.

3. Linguagem de definição e manipulação de banco de dados

Linguagem SQL padrão SQL-92

A SQL (Structured Query Language) era originariamente chamada SEQUEL (Structured English QUERY Language), quando foi implementada em um protótipo de SGBD da IBM em 1970. Num esforço conjunto do ANSI (American National Standard Institute) e da ISO (International Standards Organization), a SQL foi adotada como padrão para banco de dados relacionais. Em 1992, foi desenvolvido o padrão hoje em vigor, chamado de SQL-2 ou SQL-92, incorporando as novas características presentes nos SGBD comerciais.

A SQL é uma linguagem ampla e de difícil utilização por usuários finais. Possui comandos para definição, consulta e modificação de dados; portanto engloba tanto a DDL como a DML. Além disto, tem facilidades para definição de visões e índices, bem como para embutir comandos SQL em programas escritos numa linguagem de programação. Costuma-se dizer que cada SGBD implementa um "superconjunto de um subconjunto" da SQL-92, já que nenhum deles suporta todas as suas características e, ao mesmo tempo, cada SGBD tem suas próprias extensões não previstas da SQL-92.

Comandos em SQL-92 para definição de dados (DDL)

Esquemas em SQL

A SQL-92 usa o conceito de *catálogo*, que contém uma coleção de esquemas, formando uma esquema de informação. Este contém dados sobre todos os elementos do esquema (tabelas, visões, domínios e autorizações) que compartilham o mesmo ambiente SQL. A sintaxe dos comandos de criação e destruição de esquemas é a seguinte:

```
CREATE SCHEMA <esquema> AUTHORIZATION <usuário>
DROP SCHEMA <esquema> ( CASCADE | RESTRICT )
```

As opções CASCADE | RESTRICT indicam, respectivamente, se o dono do esquema deseja destruir o esquema com todos os seus elementos ou somente se ele não contiver nenhum elemento.

Domínios em SQL

Um domínio em SQL tem por propósito especificar um tipo de dados baseado em um tipo primitivo de dados. Por exemplo, poderíamos criar um domínio TELEFONES CHAR(10) que seria definido uma única vez e depois compartilhado por várias colunas, em várias tabelas, melhorando a legibilidade e facilitando as mudanças. Uma limitação do domínio em SQL é que ele não pode ser usado para criar novos tipos de dados definidos pelo usuário. A sintaxe dos comandos de definição de domínios é a seguinte:

```
CREATE DOMAIN <domínio> <tipo de dado primitivo>
    [ DEFAULT <valor default> ]
    [ <lista de restrições de domínio> ]
DROP DOMAIN <domínio> ( CASCADE | RESTRICT )
```

OBS.: Dados primitivos suportados pela SQL-92 (SGBD comerciais apresentam maiores variações, embora os tipos básicos sejam os mesmos):

- números inteiros (INTEGER, INT ou SMALLINT) e reais (FLOAT, REAL, DOUBLE, DECIMAL);
- cadeia de caracteres de tamanho fixo (CHAR(n));
- cadeia de caracteres de tamanho variável (VARCHAR(n));
- data e tempo (DATE, TIME, TIMESTAMP, INTERVAL).

O valor default é assumido por toda coluna definida sobre o domínio que não tiver um valor explicitamente atribuído. A lista de restrições de domínio especifica as restrições de integridade. Um exemplo simples de criação de domínio é:

```
CREATE DOMAIN TIMES CHAR(10)
    DEFAULT 'Vasco'
    CONSTRAINT TIMES_RJ
    CHECK ( VALUE IN ( 'Vasco', 'Botafogo', 'Fluminense', 'Flamengo' ))
```

A partir daí, o domínio TIMES pode ser utilizado para especificar uma coluna, cujos valores estarão restritos aos valores da lista. O nome da restrição TIMES_RJ será usado em mensagens de erro em operações de modificação da coluna. Observe que o tamanho definido para o domínio deverá ser igual ao maior valor previsto para o campo.

Tabelas e Restrições de Integridade

O conceito tabela em SQL corresponde ao de relação no Modelo Relacional. Os comandos de criação, alteração e destruição de tabelas tem a seguinte sintaxe:

```
CREATE TABLE <tabela>
    ('(<coluna> <tipo da coluna> [restrição do atributo],
     { <coluna> <tipo da coluna> [restrição do atributo] }
     [<restrição da tabela> , {<restrição da tabela>}] ')
ALTER TABLE <tabela>
    <lista de alterações na tabela>
DROP TABLE <tabela> ( CASCADE | RESTRICT )
```

A <tabela> deve pertencer a um esquema previamente criado. O nome <coluna> deve ser único dentro da mesma tabela e <tipo da coluna> pode ser um tipo primitivo ou um domínio previamente criado. Uma <restrição do atributo> pode ser usada para especificar que o atributo não pode ter valor nulo (NOT NULL) ou para atribuir um valor default (DEFAULT <valor>).

Uma <restrição da tabela> pode ser usada para especificar a chave primária (PRIMARY KEY), chaves alternativas (UNIQUE), chaves estrangeiras (FOREIGN KEY) e restrições de valores válidos (CHECK). Cada opção pode, opcionalmente, ser precedida pela cláusula CONSTRAINT <restrição>, o que permite futuras referências para alterações. Exemplo:

```
CREATE TABLE PESSOA      CREATE TABLE CONTA
(                          (Numero INTEGER NOT NULL,
CPF CHAR(11) NOT NULL,   Tipo CHAR(1) NOT NULL DEFAULT 'C'
Nome VARCHAR(50) NOT    Data_abertura DATE,
NULL,                    Limite_Credito DECIMAL (10,2),
Endereco VARCHAR(50),   CPF_Cliente CHAR(11),
Data_nascimento DATE,   PRIMARY KEY (Numero),
Renda DECIMAL(10,2),    FOREIGN KEY (CPF_cliente)
PRIMARY KEY (CPF),      REFERENCES PESSOA
UNIQUE (Nome)           ON DELETE SET NULL
);                       ON UPDATE CASCADE,
                          CONSTRAINT limite_valido
                          CHECK (Limite_credito >= 0 AND Limite_credito <= 50000));
```

O exemplo ilustra as seguintes restrições de integridade:

1. restrição de domínios (tipo da coluna e cláusula CHECK);
2. restrição de chaves (cláusulas UNIQUE e PRIMARY KEY);
3. restrição de integridade de entidades (cláusula NOT NULL para a chave primária); e
4. restrição de integridade referencial (cláusula FOREIGN KEY).

Esta última definição (4) possui a seguinte forma:

```
FOREIGN KEY ( <lista de colunas> )
    REFERENCES <tabela referenciada>
    [ ( <lista de colunas> ) ]
    [ ON DELETE <opção> ]
    [ ON UPDATE <opção> ]
```

Em geral uma chave estrangeira referencia uma chave primária da tabela referenciada. As cláusulas ON DELETE e ON UPDATE especificam uma ação corretiva caso ocorra uma alteração no valor da chave referenciada; se forem omitidas, normalmente o SGBD restringe a modificação, exibindo uma mensagem de erro alertando ao usuário da existência de valores de chave estrangeira que referenciam a chave modificada. No exemplo, se uma pessoa que possui contas for excluída da tabela PESSOA, suas contas terão o valor da coluna CPF_cliente alterado para NULL, e se o seu CPF for alterado na tabela PESSOA, todas as suas contas terão o campo CPF_CLIENTE também alterado. É importante ressaltar que as tabelas referenciadas devem ser criadas antes das tabelas com chaves estrangeiras que a referenciam. Uma opção estratégica é a especificação das chaves estrangeiras usando o comando ALTER TABLE, após a criação de todas as tabelas.

No comando ALTER TABLE, a <lista de alterações na tabela> inclui:

1. a adição de uma nova coluna;
2. a definição de um novo valor default para uma coluna existente;
3. a exclusão de uma coluna;
4. a especificação de uma nova restrição de integridade (UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK); ou
5. a exclusão de uma restrição existente.

O comando DROP TABLE destrói uma tabela, com as opções de restringir a operação se a tabela for referenciada em outra tabela ou visão, ou de remover em cascata todas as tabelas ou visões que a referenciam, juntamente com os dados armazenados.

Visões em SQL-92

Uma visão é simplesmente uma tabela derivada de outras tabelas por meio de uma consulta ao banco de dados. É uma forma limitada de visão em banco de dados, que pode ter várias tabelas formando um subconjunto do banco de dados que interessa a um usuário ou grupo de usuários.

As visões podem ser consideradas como tabelas virtuais, pois não existem em forma armazenada no banco de dados. Desta forma, conclui-se que a atualização de dados em visões é um evento restrito enquanto que não há nenhuma limitação sobre as consultas em visões. Uma vez criada a visão, ela pode ser consultada exatamente como uma tabela. A sintaxe do comando para criação de visões é a seguinte:

```
CREATE VIEW <visão> [ (coluna> {, <coluna>} ) ]  
AS <comando SELECT>
```

Como exemplo de criação de visão, o comando a seguir cria uma visão com nome, renda e limite de crédito de todas as pessoas que possuem contas do tipo “C”.

```
CREATE VIEW contas_correntes  
    (Nome, Renda, Limite_Credito)  
AS SELECT Nome, Renda, Limite_Credito  
    FROM PESSOA, CONTA  
    WHERE CPF = CPF_Cliente AND TIPO = “C”
```

Comandos em SQL-92 para manipulação de dados (DML)

Os comandos de manipulação de dados da SQL, são basicamente os de modificação de dados (INSERT, UPDATE e DELETE) e o de consulta (SELECT). Tais comandos podem ser processados de forma interativa pelos usuários ou de forma embutida em programas de aplicação. Trataremos, aqui, apenas da forma interativa.

Consultas em SQL-92

O comando básico para consultas em SQL é o SELECT. O resultado de uma consulta pode ser uma tabela com linhas duplicadas, se não houver uma chave na lista das consultadas. Em consequência, uma tabela em SQL pode não ser exatamente como uma tabela relacional que, como sabemos, é um conjunto de tuplas distintas. A cláusula DISTINCT elimina as tuplas duplicadas. A sintaxe do comando SELECT é, geralmente, a seguinte:

```
SELECT [ DISTINCT ] <lista de atributos>
      FROM <lista de tabelas ou visões>
      [ WHERE <condição de seleção/junção> ]
      [ GROUP BY <lista de atributos>
        [ HAVING <condição de seleção> ] ]
      [ ORDER BY <coluna> [ (ASC | DESC) ]
        {, <coluna> [ (ASC | DESC) ] }
```

O comando SELECT abre a possibilidade de inúmeros tipos de consultas, inclusive com comandos aninhados, isto é, comandos SELECT dentro da cláusula WHERE. Além disso, na <lista de atributos> e na <condição de seleção/junção> podem ser especificadas expressões usando funções de agregação como soma (SUM), média (AVERAGE ou AVG), contagem (COUNT), valor mínimo (MIN), valor máximo (MAX), entre outras dependendo do SGBD. Mostramos a seguir alguns exemplos auto-explicativos. É importante observar que, sempre que uma consulta envolve mais de uma tabela, uma condição de junção precisa ser especificada pois, do contrário, o resultado será um conjunto cartesiano entre as tabelas e não apenas um conjunto de tuplas que satisfazem a condição de junção.

```
SELECT * FROM PESSOA; ( * significa todos os atributos da tabela )
```

```
SELECT Nome, Endereco, Data_nascimento
FROM PESSOA
WHERE Renda > 500;
```

```
SELECT Nome, Limite_credito
FROM contas_correntes;
```

```
SELECT Nome, Tipo, Numero
FROM PESSOA, CONTA
WHERE CPF = CPF_Cliente
```

```
SELECT MAX(Limite_credito)
FROM CONTA
WHERE Tipo = 'C';
```

```
SELECT P.Nome, P.Endereco, P.Renda, Limite_credito
FROM PESSOA P, CONTA C
WHERE P.CPF = C.CPF_cliente AND C.Tipo = 'C'
ORDER BY C.Limite_credito DESC;
```

Modificando Dados em SQL-92

As operações básicas de modificação de dados são INSERT, UPDATE e DELETE, cujas sintaxes na SQL são as seguintes:

```
INSERT INTO <tabela> [ ( <lista de colunas> ) ]  
    ( VALUES ( <lista de valores> ) | <comando SELECT> )
```

```
UPDATE <tabela>  
    SET <coluna> = <valor>  
        {, <coluna> = <valor> }  
    [ WHERE <condição de seleção > ]
```

```
DELETE FROM <tabela>  
    [ WHERE <condição de seleção > ]
```

Exemplos:

```
INSERT INTO PESSOA  
    VALUES ( '444444444-00', 'João Macedo', 'Rua D No. 40 – Centro', 03/03/1972, 800);
```

```
INSERT INTO CONTA (Numero, Tipo, CPF_Cliente)  
    VALUES (00044, 'P', '444444444-00');
```

```
UPDATE CONTA  
    SET Data_abertura = 02/04/1996  
    WHERE Numero = 00044;
```

```
DELETE FROM CONTA  
    WHERE CPF_Cliente='111111111-00'
```

Índices

A linguagem SQL, além de permitir a definição de dados através da criação de esquemas, domínios, tabelas e visões, permite também a definição de índices para acelerar a consulta por colunas e grupos de colunas freqüentemente usados como argumentos de pesquisa. Um índice nada mais é do que um arquivo auxiliar, em geral implementado em múltiplos níveis como uma árvore-B, cuja única finalidade é acelerar a consulta. Em compensação, cada atualização nas tabelas indexadas que possa alterar um índice requer uma atualização do índice afetado.

```
CREATE [UNIQUE] INDEX <índice>  
    ON <tabela> ( <coluna> [ <ordem> ]  
        {, <coluna> [ <ordem> ] } )
```

```
DROP INDEX <índice>
```

Exemplos:

```
CREATE INDEX Indice_renda  
    ON PESSOA (Renda DESC, Nome ASC);
```

```
CREATE UNIQUE INDEX Indice_nome  
    ON PESSOA (Nome);
```

```
DROP INDEX Indice_renda;
```

Autorização e Controle de Acesso

A SQL possui também comandos de segurança de acesso GRANT e REVOKE, como nos exemplos auto-ilustrativos a seguir:

```
GRANT CREATETAB ON DATABASE TO user1;  
GRANT INSERT, DELETE ON PESSOA, CONTA TO user2;  
GRANT SELECT ON contas_corrente TO PUBLIC;  
GRANT CONTROL ON INDEX Indice_renda TO user1, user2;  
GRANT ALL PRIVILEGES ON PESSOA TO user1;  
REVOKE CONTROL ON INDEX Indice_renda TO user1;  
REVOKE DELETE ON PESSOA TO user2;  
REVOKE ALL PRIVILEGES ON PESSOA TO user1;
```

Controle de Transações

A SQL-92 possui comandos que permitem o controle do resultado de transações. Uma transação é uma seqüência atômica de operações do Banco de Dados que constitui a unidade lógica de trabalho dos SGBD. Na SQL não existe um comando de início da transação; é implícito que toda a operação que modifica um banco de dados inicia uma transação, que pode terminar normalmente (COMMIT) ou ser desfeita até o ponto inicial da transação (ROLLBACK).

Sempre que o comando COMMIT é executado, a transação em andamento é terminada implicitamente, as alterações realizadas pela transação são tornadas permanentes e irreversíveis, e os possíveis bloqueios que a transação mantinha são liberados. Além disto, uma nova transação é iniciada.

Quando o comando ROLLBACK é executado, as alterações realizadas pela transação em andamento, são desfeitas como se nunca tivessem ocorrido, os bloqueios são liberados e uma nova transação é implicitamente iniciada.

Em muitas implementações da SQL, existe o comando SAVEPOINT (ou CHECKPOINT) para permitir a subdivisão lógica de transações longas, como no exemplo a seguir:

```
INSERT ...  
UPDATE ...  
DELETE ...  
...  
SAVEPOINT sp_um  
INSERT ...  
UPDATE ...  
DELETE ...  
...  
SAVEPOINT sp_dois  
INSERT ...  
UPDATE ...  
DELETE ...  
...
```

A operação de ROLLBACK pode ser então realizada em subdivisões da transação, mantendo as demais partes intactas e a transação ainda ativa:

```
ROLLBACK TO SAVEPOINT sp_dois
```

4. Controles operacionais de banco de dados

Uma transação é uma unidade de execução de um programa que acessa e, possivelmente, atualiza vários itens de dados. Normalmente considera-se que um conjunto de várias operações no banco de dados é uma única unidade no ponto de vista do usuário.

Seja uma “transação” bancária onde um cliente deseja transferir um valor da conta A para a conta B. Basicamente, ele deve fornecer as seguintes informações: Conta do saque, Conta para depósito, Valor. Vamos definir o pseudo-código das operações necessárias:

Início

Validar contas

Checar saldo na conta A

Subtrair valor do saldo atual da conta A

Somar valor ao saldo da conta B

Fim

Deve ser garantido que a transferência seja totalmente realizada ou que as contas permaneçam como estão. Na visão do usuário, uma transação é uma operação única e indivisível sobre um banco de dados. Para a aplicação isso corresponde, geralmente, à execução de um conjunto de diversas operações sobre o banco de dados. Assim, os sistemas de bancos de dados devem garantir que todo este conjunto de operações seja realmente uma única unidade: ou é totalmente executado ou não é. Coleções de operações que formam uma única unidade lógica de trabalho são chamadas de *transações*.

CONCEITO:

Uma *transação* é uma unidade de programa que acessa e/ou atualiza vários itens de dados. Uma transação consiste de todas as operações executadas entre comandos do tipo “begin transaction” e “end transaction”, presentes nas linguagens de programação utilizadas.

Exemplos: Transferência de recursos de uma conta bancária para outra, marcação de assento em um voo.

Para garantir integridade de dados, os SGBD devem manter as seguintes propriedades para as transações, denominadas propriedades ACID:

Atomicidade (A): ou se executa todas as operações de uma transação ou nenhuma delas.

Consistência (C): toda transação deve preservar a consistência do banco de dados.

Isolamento (I): mesmo com a execução concorrente de transações, o efeito de cada transação deve ser o mesmo como se ela estivesse executando sozinha.

Durabilidade (D): após o término com sucesso da transação, as atualizações realizadas devem ser persistentes.

Exemplo: Seja uma transação T_i que transfere R\$ 50 de uma conta bancária A para outra conta B.

read (A);

$A := A - 50$;

write (A);

read (B);

$B := B + 50$;

write (B).

Se as contas A e B tinham inicialmente os valores R\$ 1000 e R\$ 2000 respectivamente, ao final da transação deveriam ter R\$ 950 e R\$ 2050. E se ocorrer alguma falha logo após a operação write (A)? Vamos agora considerar as propriedades ACID.

Para garantir a propriedade de *atomicidade*, todas as operações realizadas até o ponto da falha devem ser desfeitas. Caso contrário, teríamos uma *inconsistência* do banco de dados. O banco de dados tem de estar consistente antes do início de uma transação e após seu término. É claro que durante a execução de uma transação (depois de iniciada e antes de seu término), o banco de dados pode estar *temporariamente* inconsistente. No caso de *atomicidade*, esta propriedade deve ser garantida pelo SGBD, através do componente “*Gerência de Transação*”.

No caso de *consistência*, esta propriedade deve ser garantida pela aplicação, podendo ser facilitada pelas restrições de integridade ou triggers.

No caso de *durabilidade*, após o término da transação e notificação ao usuário que a mesma foi concluída com sucesso, o SGBD tem de garantir a persistência dos dados. Para garantir durabilidade:

- Os dados alterados por uma transação tem de ser gravados em disco antes do término da transação;
- O SGBD deve reconstruir as atualizações realizadas quando o sistema for restabelecido após a falha.

Durabilidade tem de ser garantida pelo SGBD, através do componente “*Gerência de Recuperação*”.

No caso de *isolamento*, o que tem de ser garantido é que mesmo com a execução concorrente de transações, o efeito de cada transação deve ser o mesmo como se ela estivesse executando sozinha. Uma solução seria executar as transações *serialmente*, sem permitir concorrência → o problema seria a *performance*. O Isolamento tem de ser garantido pelo SGBD, através do componente “*Controle de Concorrência*”.

Estados das Transações

Se durante a execução de uma transação ocorrer uma falha, a transação é “*abortada*”. Neste caso, para garantir a atomicidade, todas as operações executadas até então devem ser desfeitas, ou seja, deve ocorrer um “*rollback*” da transação, que é controlada pela “*Gerência de Recuperação*”.

Quando uma transação teve sua execução normal, deve ser “*committed*”. Uma transação committed não pode ter suas atualizações desfeitas pelo SGBD, mas apenas por uma transação chamada de *compensatória*, de responsabilidade do usuário. Uma transação deve estar em um dos seguintes estados:

- **Ativada:** fica neste estado desde o início até terminar sua execução.
- **Parcialmente Committed:** após o último comando ter sido executado.
- **Falhada:** após a descoberta de alguma anormalidade.
- **Abortada:** após a transação ter sido “rolled back”.
- **Committed:** após o término normal.

Observações:

1. Uma transação está terminada se ela foi committed ou abortada.
2. O SGBD pode se recuperar de uma falha se os dados estiverem gravados em disco. No caso de perda do disco, podem ser necessários outros recursos.
3. Uma transação em estado de abortada, pode ser reiniciada pelo sistema (se a falha foi devida a hardware ou software) ou ser eliminada (no caso de erro lógico).
4. Cuidados devem ser tomados com aplicações que emitam comprovante para o usuário. Somente devem ser emitidos após a transação estar committed.

Implementação de Atomicidade e Durabilidade

O componente “Gerência de Recuperação” de um SGBD implementa suporte para atomicidade e durabilidade. Uma solução simples para a garantia de atomicidade e durabilidade é implementada através de cópia do banco de dados.

Antes de cada transação atualizar o banco de dados, uma cópia do mesmo é realizada e as alterações são realizadas sobre a cópia. Se a transação terminou normalmente, a cópia passa a ser o novo estado do banco de dados. Caso contrário, ela é simplesmente eliminada.

Esta solução é muito ineficiente para grandes bancos de dados além de não permitir concorrência.

Execuções Concorrentes

A motivação para uso de concorrência em banco de dados é a mesma usada em multiprogramação em sistemas operacionais, ou seja, permitir executar mais transações no mesmo espaço de tempo. A questão chave é garantir consistência mesmo com a execução concorrente de transações. O problema é que mesmo que cada transação individual esteja correta, na execução concorrente a consistência pode não ser preservada.

O SGBD deve controlar a interação entre as transações concorrentes para manter a consistência. Existem diversos mecanismos para controle de concorrência.

Exemplo: Sejam as transações T1 e T2 abaixo:

Transação T1

```
read (A);  
A := A - 50;  
write (A);  
read (B);  
B := B + 50;  
write (B).  
write (B).
```

Transação T2

```
read (A);  
temp := A * 0.1;  
A := A - temp;  
write (A);  
read (B);  
B := B + temp;
```

Considerar que as contas A e B tenham, inicialmente, os valores R\$ 1000 e R\$ 2000 respectivamente.

Transação T1

```
read (A);  
A := A - 50;  
write (A);  
read (B);  
B := B + 50;  
write (B).
```

Transação T2

```
read (A);  
temp := A * 0.1;  
A := A - temp  
write (A);  
read (B);  
B := B + temp;  
write (B).
```

Escalonamento 1: Escalonamento serial de T1 seguido por T2.

Transação T1

```
write (B).  
read (A);  
A := A - 50;  
write (A);  
read (B);  
B := B + 50;  
write(B).
```

Transação T2

```
read (A);  
temp := A * 0.1;  
A := A - temp;  
write (A);  
read (B);  
B := B + temp;
```

Escalonamento 2: Escalonamento serial de T2 seguido por T1.

Um escalonamento descreve uma seqüência de execução, ou seja, a ordem cronológica em que as instruções são executadas. Os escalonamentos 1 e 2 são chamados de *serial*. Quando diversas transações são executadas concorrentemente, o escalonamento correspondente pode não ser necessariamente serial. O problema é que nem todos escalonamentos não seriais levam o banco de dados a um estado consistente.

Se o controle de concorrência for deixado inteiramente para o sistema operacional, este poderia utilizar escalonamentos que deixariam o banco de dados em um estado inconsistente. Para isso, o componente Controle de Concorrência do SGBD garante a utilização somente de escalonamentos que deixam o banco de dados consistente.

Recuperabilidade

Como vimos, se uma transação falha, é necessário desfazer todos seus efeitos para garantir a propriedade de atomicidade. Assim, em um sistema com execução concorrente, se uma transação T1 falha, é necessário que todas as outras transações que dependam de T1 sejam abortadas. Para garantir isso, é necessário estabelecer restrições sobre o tipo de escalonamentos que serão permitidos pelo sistema.

Implementação de Isolamento

Existem vários esquemas de controle de concorrência que podem ser usados para garantir a execução de transações concorrentes, somente gerando escalonamentos aceitáveis. Um exemplo trivial, seria uma transação aplicar um bloqueio (“*lock*”) sobre todo ou parte do banco de dados antes de seu início e só liberar após estar committed.

Controle de recuperação de falhas

Qualquer sistema está sujeito a falhas. No caso específico de SGBD, devem existir ações de recuperação para garantir as propriedades das transações, mesmo em caso de falhas. Vamos considerar os seguintes tipos de falhas:

- Falha na Transação
- Falha do Sistema
- Falha no Disco

As falhas de transação podem ser devidas a erros lógicos ou a erros do sistema. As falhas dos sistema podem ser devidas a problemas do hardware, do SGBD (software), do sistema operacional, alimentação elétrica etc. As falhas no disco podem ser devidas a “head crash” ou na própria operação de transferência. Para determinar como o sistema irá se recuperar de falhas, é necessário identificar que falha ocorreu. Os algoritmos de recuperação trabalham em duas questões:

1. armazenar informações durante a execução normal das transações para garantir a recuperação de falhas;
2. se uma falha ocorre, recuperar o conteúdo do banco de dados para garantir as propriedades de consistência, atomicidade e durabilidade.

Estrutura de Armazenamento

Existem, basicamente, dois tipos de armazenamento: volátil e não volátil. No tipo volátil, muito embora com acesso rápido, as informações não sobrevivem a falhas do sistema. No tipo não volátil, as informações podem sobreviver a falhas do sistema, mas o acesso é mais lento.

Devem existir outros mecanismos associados para garantir a persistência dos dados (backup, cópias de segurança etc.)

Acesso aos Dados

As informações armazenadas nos bancos de dados residem permanentemente em dispositivos não voláteis (geralmente discos). As informações são particionadas e armazenadas em unidades de tamanho fixo chamadas de blocos, que são as unidades de transferência do disco e para o disco. Blocos residentes em discos são chamados blocos físicos e os blocos residentes temporariamente na memória principal são chamados de blocos de buffer. Blocos são movimentados entre o disco e a memória principal através de duas operações:

1. input: transfere o bloco físico para a memória;
2. output: transfere o bloco de buffer para o disco.

Cada transação tem uma área privativa de trabalho, criada no início da transação, na qual cópias de todos os dados por ela acessados e atualizados são mantidos. Uma transação interage com o SGBD transferindo dados da sua área de trabalho para o buffer do sistema e vice-versa. A transferência de dados entre a transação e o SGBD é realizada pelas operações:

- read(X): lê o valor do dado X para a variável local.

Se o bloco onde X se encontra não está na memória, é realizado, primeiro, um input.

- write(X): grava o valor do dado X no bloco de buffer.

Se o bloco onde X se encontra não está na memória, é realizado, primeiro, um input.

As duas transações podem requerer a transferência de blocos do disco para a memória. Mas a transferência de dados da memória para o disco depende de outros fatores.

Recuperação e Atomicidade

Nem sempre uma operação write(X) é seguida de uma output(X), pois pode ser que o bloco possua outros dados que ainda estão sendo utilizados. O que pode ocorrer é que no caso de uma falha, o novo valor de X que ainda não foi gravado em disco será perdido. Se a transação que alterou X também modificou outros dados em outros blocos, pode ser que alguns destes já estejam em disco, o que poderá levar o banco de dados a um estado inconsistente. O que se deseja é que ou todas as alterações feitas por uma transação já *committed* sejam gravadas em disco, ou nenhuma delas. Neste caso, a propriedade de atomicidade, poderia ser garantida das seguintes formas:

- Desfazer todas as alterações parciais da transação que já tiverem sido gravadas em disco;
- Garantir a gravação em disco dos blocos que ainda estavam em memória na hora da falha.

O método mais conhecido para se implementar estes requisitos é através de *log*.

Recuperação Baseada em LOG

Log é a estrutura mais utilizada para gravar as modificações de banco de dados. É uma seqüência de registros (registros de *log*) que mantém informações sobre cada atividade de atualização do banco de dados.

Quando uma transação executa um *write*, é essencial que o registro de *log* correspondente seja criado antes da modificação do banco de dados.

Através do registro de *log* podemos modificar o banco de dados, ou até mesmo desfazer alterações já realizadas.

Registros de *log* para serem úteis na recuperação de falhas do sistema, devem estar armazenados em memória não volátil.

O *log* contém um registro completo de todas as atividades do banco de dados.

Modificação do BD Adiada

A técnica de modificação adiada garante a atomicidade da transação registrando as modificações do banco de dados no *log*, mas adiando as operações de gravação até a transação estar parcialmente *committed*. Em seguida, a informação no *log* associada com a transação é usada na execução da gravação adiada. Se ocorrer uma falha antes do *commit*, a informação do *log* é simplesmente ignorada.

Modificação Imediata do BD

Nesta técnica, as modificações no banco de dados ocorrem durante a execução da transação. Modificações gravadas por transações ativas são chamadas de modificações não *committed*. Se uma falha ocorrer durante a execução da transação, o sistema deve usar os valores antigos dos dados para restaurar o banco de dados, ou seja, voltar ao estado anterior ao início da transação.

Checkpoints

Quando uma falha ocorre, a princípio o sistema deveria consultar todo o *log* para determinar quais transações necessitam ser desfeitas e quais necessitam ser refeitas. Tal necessidade leva a dois problemas:

1. Tempo no processo de pesquisa do *log*;
2. Algumas transações já tinham gravado seus dados no banco de dados.

Para reduzir este *overhead*, foi introduzido o conceito de *checkpoints*. Durante a execução, o sistema mantém o *log* usando uma das duas técnicas vistas anteriormente. Além disso, o sistema estabelece os *checkpoints* que requerem a seguinte seqüência de operações:

1. Gravar em disco, todos os registros de *log* residentes na memória principal;
2. Gravar em disco, todos os blocos de *buffer* modificados;
3. Gravar em disco um registro de *log* do tipo <checkpoint>.

Quando ocorre uma falha, basta o sistema percorrer o arquivo de *log* para trás até encontrar o primeiro registro de *checkpoint*. Todas as alterações realizadas pelas transações já *committed* antes do *checkpoint* estarão no banco de dados.

Recuperação com Transações Concorrentes

Neste caso, a recuperação é mais complexa. Até porque, independente do número de transações concorrentes, o sistema só tem um *buffer* e um arquivo de *log*. O esquema de recuperação depende fortemente do esquema de controle de concorrência utilizado. Por exemplo, para fazer *rollback* de uma transação falhada, as alterações por ela realizadas devem ser desfeitas, bastando para isso fazer a operação *undo*.

O problema é se outra transação leu um dado não *committed*. Como desfazê-la? No caso de não haver leitura de dado não *committed*, para recuperação de uma transação falhada, basta

pesquisar para trás no arquivo de *log* todos os registros daquela transação específica e desfazer as alterações.

Considerando agora a concorrência, é necessário, em cada registro de *checkpoint* do *log*, incluir todas as transações ativas no momento do *checkpoint* (pode haver mais de uma). Na recuperação de uma falha, o sistema constrói duas listas, lista-redo e lista-undo, contendo as transações que devem ser refeitas ou desfeitas respectivamente. Pesquisando o *log* para trás até encontrar o *checkpoint*, todas as transações com registro de *log* $\langle T \text{ commit} \rangle$ são incluídas na lista-redo. Todas as transações com registro de *log* $\langle T \text{ start} \rangle$, que não estiverem na lista-redo, serão incluídas na lista-undo. Em seguida, é necessário incluir as transações que estavam ativas na hora do *checkpoint*. As que não estiverem na lista-redo são incluídas na lista-undo. Depois, basta percorrer primeiro a lista-undo, a partir da transação mais recente para trás, e após percorrer a lista-redo de trás para frente.

Controle de concorrência

Em execução concorrente de transações, a propriedade de isolamento pode não ser preservada. Torna-se necessário o controle da interação entre estas transações, através de mecanismos de *Controle de Concorrência*. Um modo de garantir seriabilidade é garantir que enquanto uma transação está acessando um dado, nenhuma outra possa modificá-lo. A solução mais comum é através de bloqueio (“*lock*”).

Protocolos Baseados em Lock

Existem várias maneiras de se fazer *lock* sobre dados. Podemos, inicialmente, definir os seguintes tipos de *lock*:

- **Shared (S)**: quando a transação quer apenas LER o dado;
- **Exclusive (X)**: quando a transação necessita GRAVAR o dado.

Transações devem requisitar *lock* através das instruções **lock-S(Q)** ou **lock-X(Q)**. Uma requisição pode ser negada pela gerência de controle de concorrência se o item de dado em questão já estiver *locked* por outra transação em um modo incompatível. Para melhorar a performance, é desejável que a transação faça *unlock* dos dados o mais cedo possível.

O uso de *locking* pode produzir *deadlock*. Quando ocorre *deadlock*, o sistema deve fazer *rollback* das transações envolvidas, e os dados destas tornam-se *unlocked*. *Deadlocks* são preferíveis a estados inconsistentes, pois podem ser tratados pelos SGBD. As Transações devem seguir um conjunto de regras, chamado de *protocolo de locking*, definindo quando pode fazer *lock* e *unlock*. Os protocolos de *locking* restringem o número de escalonamentos possíveis, mas todos serão serializáveis.

Uma questão a ser observada na concessão (*grant*) de *locks* é a possibilidade de uma transação **T** solicitar um **lock-X(A)** e não conseguir por existir uma outra transação **T1** do tipo **lock-S(A)** executando. Se novas transações **lock-S(A)** chegam, estas poderiam ser executadas antes de **T** pois não haveria conflito com **T1**. Assim, **T** poderia ter que esperar muito tempo. Uma maneira de evitar esse problema é utilizar o seguinte procedimento para conceder *lock*:

- Não existir outra transação que já tenha *lock* sobre o dado, de forma conflitante;
- Não existir outra transação que já esteja esperando pelo *unlock* do dado.

Protocolo Two-Phase Locking

Este protocolo garante seriabilidade e requer que cada transação requisite *lock* e *unlock* em duas fases:

- Fase de crescimento: A transação somente pode obter *locks*;
- Fase de encolhimento: A transação apenas pode liberar *locks*.

Inicialmente a transação está na fase de crescimento e quando libera o primeiro lock entra automaticamente na fase de encolhimento.

O ponto no escalonamento onde a transação obteve seu último *lock* (fim da fase de crescimento) é chamado de *lock point*. Ordenando as transações por *lock points*, é garantida a seriabilidade quanto ao conflito. Um problema deste protocolo é que permite a ocorrência de *deadlock* e de *rollback* em cascata.

Rollback em cascata pode ser evitado com o uso do protocolo strict two-phase locking. Neste protocolo, todos os *locks* exclusivos só são liberados quando a transação termina (commit). Um outro protocolo é o rigorous two-phase locking, onde todos os *locks* só são liberados quando a transação termina (commit). A maioria dos SGBDs implementa um destes protocolos.

Protocolos Baseados em Timestamp

Para cada transação **T** no sistema é atribuído um *timestamp* (marcação de tempo), denotada por **TS(T)**. O *timestamp* pode ser o próprio relógio do sistema ou mesmo um contador lógico seqüencial controlado pelo SGBD. O *timestamp* de cada uma das transações determina a ordem de seriabilidade.

Para implementar este esquema, para cada item de dados **Q** são associados dois *timestamps*:

- **W-timestamp(Q)**: determina o maior timestamp de qualquer transação que executou **write(Q)** com sucesso.
- **R-timestamp(Q)**: determina o maior timestamp de qualquer transação que executou **read(Q)** com sucesso.

Estes *timestamps* são atualizados sempre que uma nova instrução **read** ou **write** é executada.

Protocolos Baseados em Validação

Este protocolo se aplica nos casos onde ocorrem poucos conflitos. Se baseia no fato de que transações participam de duas ou três fases diferentes durante sua execução:

- Fase de leitura: os valores lidos e gravados ficam armazenados em variáveis locais.
- Fase de validação: é realizada uma validação para determinar se os dados alterados localmente podem ser gravados no banco de dados.
- Fase de gravação: de acordo com a validação, os dados são gravados no banco de dados.

Para a validação, é necessário associar a cada transação três *timestamps*:

- **Start(T)**: quando T inicia execução.
- **Validation(T)**: quando termina fase de leitura e inicia validação.
- **Finish(T)**: quando termina gravação.

Granularidade Múltipla

Os esquemas de controle de concorrência discutidos até então usava *lock* sobre um dado individual. Mas, se uma transação necessita acessar todo o banco de dados, é mais conveniente fazê-lo com apenas um *lock*. Por outro lado, se ela necessitar acessar poucos dados, seria inconveniente ter de fazer *lock* em todo o banco de dados. Assim, é conveniente ter um mecanismo que permita trabalhar com vários níveis de granularidade. Uma solução é utilizar uma hierarquia de granularidade de dados, representada graficamente por uma árvore, da maior granularidade para a menor. Cada nó na árvore poderia ser bloqueado individualmente. Tal como no protocolo *two-phase locking*, são usados os modos compartilhado e exclusivo. Quando uma transação bloqueia um nó da árvore (exclusivo ou compartilhado), todos os descendentes daquele nó também ficam bloqueados.

Esquemas de Multiversão

Os esquemas de controle de concorrência discutidos até então asseguram seriabilidade, atrasando uma operação ou abortando a transação. Em um sistema de banco de dados multiversão, cada operação **write(Q)** cria uma nova versão de **Q**. Quando uma operação **read(Q)** é emitida, o sistema seleciona uma das versões para ser lida. O esquema de controle de concorrência tem de garantir que a seleção da versão a ser lida assegure seriabilidade.

Tratamento de Deadlock

Um sistema está em deadlock se existe um conjunto de transações tal que toda transação no conjunto esteja esperando por outra transação no conjunto. A única solução é o sistema desfazer algumas das transações envolvidas. Existem dois métodos principais para tratar do problema de *deadlocks*: Prevenção e Detecção e Recuperação.

Prevenção é mais freqüentemente usada se a probabilidade do sistema entrar em *deadlock* for alta. O esquema mais simples é requerer que cada transação bloqueie todos os seus dados antes de iniciar a execução. Uma desvantagem é a diminuição da concorrência e outra é que uma transação pode ter de esperar muito até conseguir bloquear todos os dados necessários. Um outro esquema é usar apropriação e desfazer transações. São utilizados *timestamps* para determinar se uma transação deve esperar ou ser desfeita. Existem dois esquemas: espere-morra (*wait-die*) e machuque-espere (*wound-wait*).

No esquema espere-morra (*wait-die*), quando uma transação **T1** requer um dado bloqueado por outra transação **T2**, **T1** somente espera se seu *timestamp* for menor que o de **T2**. Caso contrário, **T1** é desfeita (morre).

No esquema machuque-espere (*wound-wait*), quando uma transação **T1** requisita um dado bloqueado por **T2**, **T1** espera somente se seu *timestamp* for maior que o de **T2**. Caso contrário, **T2** é desfeita (**T2** é machucada por **T1**). Uma transação desfeita mantém seu *timestamp* antigo, evitando, assim, inanições.

Se *deadlocks* podem ocorrer, então um esquema de detecção e recuperação é necessário. Um algoritmo que examina o estado do sistema é invocado periodicamente para saber se um *deadlock* ocorreu. Assim, para detecção e recuperação de *deadlock*, o sistema precisa:

- Manter informações sobre dados alocados por transações;
- De um algoritmo para determinar se o sistema entrou em *deadlock*;
- Recuperar-se do *deadlock*.

Detecção de Deadlock

Deadlocks podem ser descritos em termos de um grafo direcionado chamado de grafo de espera. Um *deadlock* existe no sistema se e somente se o grafo de espera contiver um ciclo. Cada transação envolvida no ciclo está em *deadlock*. Para detectar *deadlock*, o sistema precisa manter o grafo de espera e invocar periodicamente um algoritmo que procure por um ciclo no grafo.

Recuperação de Deadlock

Quando um *deadlock* é detectado, o sistema precisa se recuperar e para isso a solução mais comum é desfazer uma ou mais transações. Para isso, três questões precisam ser consideradas:

- Selecionar uma vítima
- Desfazer transações (rollback)
- Inanição

Para selecionar uma vítima entre um conjunto de transações, o ideal é aquela que proporcione custo mínimo. Entre os fatores que determinam o custo de desfazer uma transação, estão:

- tempo de execução já ocorrido e o que falta;

- Quantos dados está utilizando;
- Quantos novos dados precisa;
- Quantas transações serão desfeitas.

Decidida a transação que será desfeita, o *rollback* é realizado e a mesma é pode ser reiniciada automaticamente pelo sistema. Em sistemas onde a seleção de vítimas é baseada em custo, pode ocorrer que a mesma transação seja sempre escolhida como vítima. É necessário assegurar que uma mesma transação seja escolhida como vítima, no máximo, um número finito de vezes. A solução é incluir o número de *rollbacks* já ocorridos na transação no fator de custo.

Operações de Inclusão e de Exclusão

Além das operações de **read** e **write**, as operações de inclusão e exclusão também afetam o controle de concorrência. Por exemplo, uma tentativa de leitura por uma transação de um dado já excluído, resulta em erro lógico. Uma tentativa de leitura por uma transação de um dado ainda não incluído, também resulta em erro lógico. Também é um erro lógico excluir um dado inexistente.

Operações de Exclusão.

No caso de uma transação **T1** que tenha uma operação de **delete(Q)**, pode haver conflito com outra transação **T2** que tenha operações de **read(Q)**, **write(Q)**, **delete(Q)** e **insert(Q)**, de acordo com a ordenação destas operações.

Uma operação de **insert(Q)** é tratada similarmente a uma de **write(Q)** para os objetivos de controle de concorrência.

Controle de segurança e integridade

Os dados armazenados no banco de dados precisam ser protegidos contra perda, acessos não autorizados, destruição ou alteração intencional e introdução acidental de inconsistência. Além das restrições de integridade, propriedades ACID, recuperação de falhas etc., é necessário dispor de outros mecanismos para garantir a consistência e durabilidade do banco de dados.

Violações de Segurança e Integridade

O mau uso do banco de dados pode ser classificado como sendo intencional (malicioso) ou acidental.

A perda acidental de consistência de dados pode resultar de:

- quebras durante o processamento da transação;
- anomalias causadas por acesso concorrente ao banco de dados;
- anomalias causadas pela distribuição dos dados em uma rede;
- um erro lógico nas transações.

É mais fácil proteger o sistema contra perdas acidentais que maldosos, como:

- leitura não autorizada de dados (roubo de informações);
- modificação não autorizada de dados;
- destruição de dados.

A segurança do banco de dados se refere a proteção contra acessos maldosos. A integridade se refere ao ato de evitar a perda acidental de consistência. Nem sempre é possível a distinção destes conceitos.

Medidas de segurança precisam ser tomadas em diversos níveis.

- No nível físico, o banco de dados (computadores) devem estar em locais seguros;
- No nível humano, os usuários devem ser cautelosamente autorizados.

Autorização e Visão

Um usuário pode ter uma ou mais formas de autorização sobre partes do banco de dados, tais como:

- autorização para ler;
- autorização para inserir;
- autorização para alterar;
- autorização para excluir.

Além destas autorizações para acessar dados, o usuário pode ter autorização para modificar o esquema, como incluir relações, índices etc.

Através do conceito de visão, é possível “personalizar” um banco de dados para um usuário específico. As definições de visão são feitas através da SQL assim como os comandos para conceder ou revogar privilégios.

5. Administração de banco de dados

Fundamentos

Entende-se por administração de banco de dados a execução de um conjunto de tarefas (por uma pessoa ou grupos de pessoas) de controle do sistema. As responsabilidades do Administrador do Banco de Dados (DBA) incluem:

- Decidir o conteúdo de informações do banco de dados, isto é, identificar as entidades do interesse da empresa e a informação a registrar em relação a estas entidades. Uma vez feito isto, o DBA deve então definir o conteúdo do banco de dados descrevendo o esquema conceitual. A forma objeto (compilado) daquele esquema é utilizada pelo SGBD para responder as solicitações de acesso. A forma (não compilada) atua como documento de referência para os usuários do sistema.
- Decidir a estrutura de armazenamento e a estratégia de acesso, isto é, decidir como os dados serão representados no banco de dados e definir esta representação escrevendo a definição da estrutura de armazenamento.
- Servir de ligação com os usuários a fim de garantir a disponibilidade dos dados que estes necessitam.
- Definir os controles de segurança e integridade, que são considerados como parte do esquema conceitual.
- Definir a estratégia de reserva e recuperação. A partir do momento em que a empresa começa efetivamente a basear-se em banco de dados, torna-se dependente do bom funcionamento deste sistema. Na eventualidade de danos ao banco de dados – causados por erro humano ou falha no hardware – é de suma importância fazer retornar os dados envolvidos com um mínimo de demora e com as menores conseqüências ao restante do sistema.
- Monitorar o desempenho e atender as necessidades de modificações, organizando o sistema de tal maneira que dele seja obtido o melhor desempenho.

Evidentemente serão necessários diversos programas utilitários como auxílio as tarefas pendentes. Listamos, a seguir, alguns exemplos dos programas utilitários necessários.

- Rotinas de carga (para criar uma versão inicial do banco de dados a partir de um ou mais arquivos);
- Rotinas de backup e recuperação (para realização de cópias de segurança e recarga do banco de dados a partir destas cópias).
- Rotinas de reorganização → para rearrumar os dados em vista de diversas razões de desempenho – por exemplo, agrupar dados de certa maneira ou regenerar espaço ocupado por dados que se tornaram obsoletos.

- Rotinas estatísticas para computar diversos desempenhos estatísticos, tamanhos de arquivos e distribuição de valores de dados.
- Rotinas analíticas, para análise das estatísticas mencionadas.

Uma das ferramentas mais importantes para administração do banco de dados é o *dicionário de dados*, conhecido também como catálogo do sistema. O dicionário de dados é um banco de dados do sistema e seu conteúdo pode ser considerado como sendo “dados sobre dados”, ou seja, descrições dos objetos do sistema ao invés de dados brutos. O dicionário de dados também pode estar integrado ao banco de dados que descreve, incluindo portanto sua própria descrição (embora tal integração denominada *dicionário de dados ativo*, geralmente, impacte na performance do sistema como um todo).

Tunning

O desempenho da maioria dos sistemas (pelo menos antes de serem ajustados) está limitado, primeiramente, pelo desempenho de um ou de uns poucos componentes que são chamados gargalos. Quando ajustamos um sistema, devemos tentar primeiro descobrir quais são os gargalos e então eliminá-los aperfeiçoando o desempenho dos componentes causadores dos gargalos. Quando um gargalo é removido, pode acontecer de um outro componente se tornar um gargalo. Em um sistema equilibrado, nenhum componente único é o gargalo.

Para programas simples, o tempo gasto em cada região do código determina o tempo de execução total. Entretanto, sistemas de bancos de dados são muito mais complexos e são modelados como sistemas em fila (queueing systems). Uma transação exige vários serviços de um banco de dados e cada um desses serviços tem uma fila a ele associada. Pequenas transações podem gastar muito de seu tempo esperando em filas. Como consequência, os gargalos são normalmente resultado de filas de espera por determinado serviço. A utilização dos recursos deve ser mantida baixa o suficiente para que o tempo de espera para sua utilização seja curto.

Os administradores de banco de dados podem ajustar o SBD em três níveis. O primeiro nível (e o mais baixo) é o de hardware. Nele as opções de ajuste incluem a adição de discos, memória ou processadores.

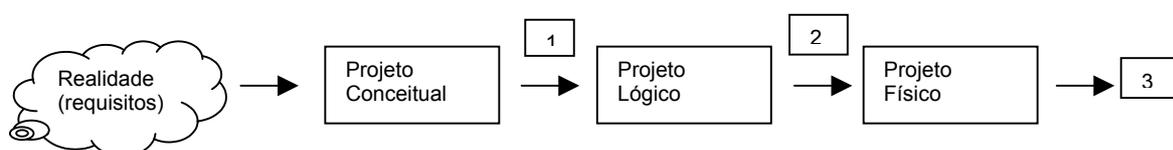
O segundo nível consiste em parâmetros (específicos) de cada sistema de banco de dados que geralmente incluem o tamanho do buffer e o controle de checkpoints.

O terceiro nível (mais alto) engloba ajustes no projeto do esquema, a criação de índices e a execução de transações. O ajuste nesse nível é relativamente independente do sistema.

Para testar o desempenho de um sistema de banco de dados mesmo antes de instalá-lo, podemos criar um modelo de simulação de desempenho onde cada um dos serviços (CPU, discos, buffer e controle de concorrência, por exemplo) será modelado.

Projeto de Banco de Dados

O projeto de um banco de dados é um processo complexo que envolve inúmeras decisões em vários níveis diferentes. Tal complexidade poderá ser melhor gerenciada se o problema for decomposto em “subproblemas” com soluções independentes e com o apoio de métodos e técnicas específicas para cada um. Podemos decompor o projeto do banco de dados em projeto conceitual, projeto lógico e projeto físico, conforme o esquema abaixo:



– 1 – Modelo Conceitual; – 2 – Modelo Lógico; – 3 – Modelo Físico

6. Projeto conceitual de banco de dados

Requisitos do projeto conceitual de dados

Dada uma coleção de requisitos de dados, resultante da coleta e análise de requisitos, o papel do projetista é criar um esquema conceitual que os satisfaça. Estes requisitos podem ser apresentados sob a forma de especificação em linguagem natural, como formulários em papel usados para preenchimento de dados ou como modelos representativos dos dados contidos em arquivos utilizados por programas de computador já existentes. Numa situação geral, vários esquemas externos são criados, pois cada usuário ou grupo de usuários pode ter diferentes visões do sistema.

Existem várias estratégias para projetar esquemas conceituais. A maioria segue uma abordagem incremental (refinamentos sucessivos), em que se parte de um conceito inicial derivado dos requisitos e sucessivamente se aplicam modificações, refinamentos ou abstrações, até se obter o esquema final. As estratégias básicas são a estratégia de projeto descendente (*top-down*) e a estratégia de projeto ascendente (*bottom-up*).

Na estratégia *top-down*, o projetista inicia com um esquema com alto nível de abstração e então aplica refinamentos sucessivos tais como a especificação de atributos, relacionamentos, especializações, agregações e novas entidades. Na estratégia *bottom-up*, o projetista inicia com um esquema contendo conceitos básicos – em geral atributos ou itens de dados dispersos nos requisitos – e então aplica combinações e extensões destes conceitos, como o agrupamento de atributos em entidades e relacionamentos, e a descoberta de novos relacionamentos, agregações e generalizações.

Existe ainda a estratégia *inside-out* onde o ponto de partida é um conjunto de conceitos mais evidentes, em geral as entidades mais relevantes do problema. O processo avança com a incorporação de novos conceitos (atributos, relacionamentos, novas entidades) na vizinhança dos conceitos já modelados. Esta estratégia é um caso especial da estratégia *bottom-up* com uma seqüência de relacionamentos mais disciplinada.

O propósito do projeto conceitual é descrever o conteúdo de informação do banco de dados ao invés das estruturas de armazenamento necessárias. Desta forma, podemos listar os seguintes requisitos independentes da tecnologia a serem utilizados na implementação, devendo nortear o projeto conceitual de qualquer banco de dados: completeza, minimalidade, correção, expressividade, legibilidade e flexibilidade.

Completeza

A estrutura de um banco de dados é completa se cada classe de objetos do mini-mundo correspondente é representado no esquema desse banco de dados. Um esquema de banco de dados é completo se atender a todos os requisitos de informação do sistema.

Minimalidade

Um esquema de banco de dados é MINIMAL se cada classe de objetos do mini-mundo correspondente é representado no esquema por no máximo uma classe de objetos de dados. Os objetos de uma parcela do mundo real (mini-mundo) devem estar representados preferencialmente através de um único objeto de dados no banco de dados. Ou seja, a redundância (repetição desnecessária) de dados deve ser evitada ao máximo e se ela for necessária, por uma questão de performance, por exemplo, devem ser criados mecanismos que a controlem de forma eficaz. A integração de dados, que é uma técnica de projeto de banco de dados, visa garantir esta característica desejável.

Correção

Um esquema de banco de dados é correto se cada classe de objetos do mini-mundo correspondente é representada no esquema pela classe de objetos de dados apropriada.

Expressividade

Um esquema de banco de dados é expressivo se cada classe de objetos do mini-mundo correspondente é representado no esquema da forma mais detalhada possível. A estrutura de um banco de dados deve ser facilmente descrita ou especificada. Ou seja, um projetista de banco de dados deve ter ferramentas que simplifiquem e facilitem a especificação da estrutura de um banco de dados. Tais ferramentas ou modelos de dados devem permitir a especificação do banco de dados em diversos níveis de abstração, sendo que em cada nível o projetista deve ter uma preocupação diferente, se abstraindo de detalhes não relevantes para aquele nível. A abstração de dados visa garantir esta característica desejável.

Legibilidade

Um esquema de banco de dados é legível quando ele representa os requisitos de modo natural, podendo ser facilmente compreendido sem a necessidade de explicações posteriores. Tal legibilidade deve ser expressa em termos gráficos e sintáticos.

Flexibilidade

Um esquema de banco de dados é flexível se ele puder se adaptar facilmente a mudanças de registros. Se o esquema de banco de dados for Correto e Minimal é provável que ele seja Flexível a mudanças. Um banco de dados deve ser flexível a mudanças estruturais na parcela do mundo real correspondente. Ou seja, um banco de dados deve suportar mudanças em sua estrutura de tal forma que isso não afete as transações que o manipulam. Em outras palavras as transações devem ser independentes de mudanças na estrutura do banco de dados. A independência de dados (lógica e física) visa garantir esta característica desejável.

Utilização do modelo entidade-relacionamento como ferramenta para o projeto conceitual

Ao se utilizar a Modelagem Conceitual de Dados com a técnica de Entidades e Relacionamentos, obteremos resultados e esquemas puramente conceituais sobre a essência de um sistema, ou melhor, sobre o negócio para o qual estamos desenvolvendo um projeto, não representando-se procedimentos ou fluxo de dados existentes.

As literaturas existentes nunca deixam claro como podemos entender entidades, atributos, generalizações, especializações e relacionamentos.

Define-se *entidades* como aquele objeto que existe no mundo real com uma identificação distinta e com um significado próprio. São as “coisas” que existem no negócio, ou ainda, descrevem o negócio em si. Se alguma “coisa” existente no negócio nos proporciona algum interesse em mantermos dados armazenados sobre ela, isto a caracteriza como uma entidade do negócio. Uma entidade será então um conjunto de dados em nosso modelo conceitual. Devemos, ao visualizar uma entidade, entendê-la como uma tabela de dados, onde cada linha desta tabela representa uma instância da mesma.

Os dados que caracterizam um objeto, são os *atributos* inerentes à entidade. Todo objeto para ser uma entidade possui propriedades que são descritas por atributos e valores. Estes atributos e seus valores, juntos, descrevem as instâncias de uma entidade. Por exemplo, em uma empresa temos um objeto sobre o qual desejamos manter dados armazenados (entidade), chamado funcionário. Um funcionário é descrito por um número de matrícula, um nome, uma data de admissão, data de nascimento, valor de seu salário, etc. Cada instância de funcionário será formada

por valores nestes atributos, sendo que é o conjunto desses valores que devemos visualizar como uma linha de uma tabela de dados (*tupla*). Os valores de determinados atributos ou de um determinado atributo nunca se repetem nas ocorrências das entidades, caracterizando que não existem objetos repetidos dentro da entidade. Estes atributos cujos valores nunca se repetem, tem a função de atuar como identificadores únicos das instâncias da entidade. Chave primária, então, é o atributo que identifica uma única ocorrência dentro de uma tabela.

Quando encontramos entidades que possuem o mesmo conjunto de atributos para descrevê-las, podemos generalizá-las em uma única entidade, mantendo sua identidade de subconjunto através da inserção de um atributo qualificador para as ocorrências de cada uma. Por exemplo, um médico pode ser um cardiologista, ou neurologista, ou pediatra, ou clínico geral, etc. A esta qualificação por atributos que nos permitirá identificar um grupo, uma classe dentro da classe genérica, denominamos de *Especialização*.

Quando existirem entidades diversas com nomes distintos, mas que na realidade podem ser generalizadas em uma única, já que conceitualmente referem-se a um macro objeto, que por generalização pode absorvê-las integralmente. Na prática, a *Generalização* é efetivada quando o conjunto de atributos das entidades é comum em sua maior parte. Concluindo, na generalização estamos colocando todas as instâncias de entidades diversas em uma única entidade, realizando seu tratamento como um todo, ou como parte quando necessário.

Um *Relacionamento* pode ser definido como o fato, o acontecimento que liga dois objetos, duas “coisas” existentes no mundo real. Este conceito pode ser estendido como o fato que efetua a junção de duas ou mais tabelas de dados.

7. Projeto lógico de banco de dados

O projeto lógico de um banco de dados relacional consiste no mapeamento do esquema conceitual para o modelo de dados do SGBD escolhido. Isto pressupõe uma fase anterior de escolha do SGBD. Uma vez conhecido o SGBD, o processo prossegue até a geração dos comandos de definição do banco de dados. Em bancos de dados centralizados, esta geração de comandos é única, enquanto que em bancos de dados distribuídos pode ser necessária uma geração de comandos para cada local onde serão armazenados os dados.

Utilização do modelo de dados relacional como ferramenta para o projeto lógico de dados.

O mapeamento que transformará o esquema conceitual em um esquema lógico, pode ser visto com um processo em duas fases: a primeira, independente do SGBD, não considera nenhuma característica específica que se aplica à implementação particular do SGBD escolhido; a segunda, consiste no ajustamento do esquema lógico às suas particularidades.

Hoje em dia, esta fase do projeto é automatizada com a utilização de ferramentas CASE. Na prática atual, a maioria dos ajustes, mesmo os físicos, como a criação de índices, domínios e regras de validação, pode ser feita diretamente nos diagramas dos esquemas lógicos; a tradução do esquema em comandos da DDL do SGBD é feita automaticamente gerando um arquivo para execução imediata ou posterior.

As regras para tradução do Diagrama de Entidades e Relacionamentos (DER) em um Diagrama de Estrutura de Dados (DED), são simples e diretas. No caso de entidades e agregações, cada conceito do DER dá origem a uma tabela no DED. Os atributos da entidade são mapeados em atributos da tabela, e o atributo identificador da entidade corresponde a chave primária da tabela derivada. No caso de relacionamentos, podem ser aplicadas as seguintes regras:

- Relacionamentos 1:1 e 1:n não originam tabelas. A representação do relacionamento no modelo relacional é feita com a inclusão na tabela derivada da entidade da entidade do *lado n* de um

atributo correspondente à chave primária da entidade do *lado 1*, com a função de chave estrangeira;

- Cada relacionamento n:m dá origem a uma tabela no DED. A sua chave primária é derivada da concatenação dos atributos identificadores das entidades relacionadas;
- Cada relacionamento com grau 3 (ternário) ou superior também dá origem a uma tabela no DED. Da mesma forma observada para os relacionamentos n:m, a sua chave primária é derivada da concatenação dos atributos identificadores das entidades relacionadas.

Uma prática comum na Análise Estruturada Moderna é a utilização de nomes de tabelas no plural, os quais corresponderão aos nomes de depósitos de dados na Análise Funcional.

Como exemplo de uma ferramenta CASE de uso comercial, o ERwin da *Logic Works* utiliza uma forma reduzida do diagrama usado na metodologia IDEFIX americana. Pode-se dizer que um diagrama gerado nesta ferramenta é um DED com informações conceituais usualmente descritas no DER (tais como cardinalidades mínimas e máximas, relacionamentos identificadores, entidades fracas) mais informações físicas que só aparecem nos comandos da DDL do SGBD alvo.

Normalização de dados.

O processo de normalização nasceu junto com o Modelo Relacional, como um mecanismo formal para analisar esquemas de relações, baseado nas suas chaves e nas dependências funcionais entre seus atributos.

O processo consiste em projetar relações normalizadas a partir de relações não normalizadas até que sejam eliminadas as possibilidades de anomalias de atualização. Considere, por exemplo, uma relação com o seguinte esquema:

PACIENTES (ID, Nome, Endereço, Telefone, Sexo, Data_nascimento, Sigla_convenio, Nome_convenio, Endereço_convenio, Telefone_convenio).

Essa relação possui uma estrutura que facilita a ocorrência das seguintes anomalias no momento da atualização dos dados:

- *Anomalia de inserção*: ao se inserir um novo paciente, se este for associado de um convênio, deve-se também inserir dados do convênio (nome, endereço, telefone) mesmo que já estejam cadastrados. Outra anomalia é que não se pode inserir um convênio sem inserir também um paciente, já que a chave primária ID seria nula;
- *Anomalia de exclusão*: ao se excluir um paciente cadastrado, se este for o único associado a um convênio, os dados sobre o convênio serão perdidos;
- *Anomalia de modificação*: ao se modificar os dados de um convênio, será necessário modificar os mesmos dados em todas as tuplas de pacientes que estejam associados àquele convênio.

Uma relação está em Primeira Forma Normal (1FN) se, e somente se, todos os seus atributos contêm apenas valores atômicos (simples, indivisíveis). Atributos multivalorados – como múltiplos telefones de pacientes – devem ser eliminados, com redundância de dados, ou criando uma nova relação apenas para armazenar os valores das diferentes localizações. Uma terceira alternativa seria criar um campo para cada valor; neste caso, um número máximo de valores seria fixado, além de dar margem a muitos valores nulos. Da mesma forma, atributos compostos devem ser eliminados, tratando cada componente como um campo de valor atômico.

Uma relação está em Segunda Forma Normal (2FN) se, e somente se, estiver em 1FN e todo atributo não-primário (isto é, que não seja membro da chave primária) for totalmente dependente da chave primária. No exemplo a seguir, a relação CONSULTAS está em 1FN porque não possui

atributos multivalorados ou compostos. Entretanto, não está em 2FN, por que existem dependências funcionais parciais em relação à chave {Id_Paciente, Id_Medico}.

CONSULTAS (Id_paciente, Id_Medico, Data, Hora, Nome_paciente, Nome_medico
CRM_medico).

No caso, as dependências funcionais são:

{Id_Paciente, Id_Medico} → {Data, Hora}

{Id_Paciente} → {Nome_paciente}

{Id_Medico} → {Nome_medico, CRM_medico}

Essas dependências significam que ID_Paciente determina Nome_paciente, e Id_Medico determina Nome_medico e CRM_medico, isto é, os atributos não-primos Nome_paciente, Nome_medico e CRM_medico não dependem totalmente da chave primária {Id_Paciente, Id_Medico}. O problema é que a relação não está bem projetada, misturando conceitos diversos (pacientes, médicos e consultas entre pacientes e médicos) numa única representação. A solução é tratar estes conceitos em relações separadas, conforme apresentado a seguir:

PACIENTES (Id_paciente, Nome_paciente)

MÉDICOS (Id_medico, Nome_medico, CRM_Medico)

CONSULTAS (Id_Paciente, Id_Medico, Data, Hora)

Uma relação está em Terceira Forma Normal (3FN) se, e somente se, estiver em 2FN e nenhum atributo não-primo for transitivamente dependente da chave primária. No exemplo a seguir, a relação PACIENTES está em 2FN porque não possui dependências parciais em relação à chave (até porque a chave é simples), porém, não está em 3FN porque existe dependência transitiva de atributo não-primo em relação à chave.

PACIENTES (Id, Nome, Endereço, Telefone, Sexo, Sigla_convenio, Nome_convenio,
Telefone_convenio)

As dependências funcionais são:

{Id} → {Nome, Endereço, Telefone, Sexo, Sigla_convenio}

{Sigla_convenio} → {Nome_convenio, Telefone_convenio}

Como podemos observar, Nome_convenio e Telefone_convenio são atributos não-primos transitivamente dependentes em relação à chave, violando a 3FN. O problema, novamente, é que conceitos diversos (pacientes e convênios) estão representados na mesma relação. De forma semelhante ao efetuado para a 2FN, a solução é também separar os conceitos de acordo com as dependências identificadas, compondo relações diferentes, conforme apresentado a seguir:

PACIENTES (Id, Nome, Endereço, Telefone, Sexo, Sigla_convenio)

CONVENIOS (Sigla_convenio, Nome_convenio, Telefone_convenio)

Para definir a *Forma Normal de Boyce-Codd (FNBC)*, é necessário introduzir o conceito de *superchave* de uma relação, que é qualquer subconjunto dos atributos cujos valores combinados não se repetem na relação. Assim, uma chave (candidata ou primária) é uma superchave, e qualquer conjunto de atributos que inclua uma chave é também uma superchave. Usando o conceito de superchave, a 3FN pode ser redefinida da seguinte forma:

Uma relação está em 3FN se para toda dependência funcional $X \rightarrow A$, for válida uma das seguintes condições: i) X é uma superchave ou ii) A é membro de uma chave candidata.

No exemplo anterior, a dependência funcional {Sigla_convenio} \rightarrow {Nome_convenio, Telefone_convenio} viola a 3FN na relação PACIENTES inicial porque {Sigla_convenio} não é superchave e {Nome_convenio, Telefone_convenio} não é membro da chave candidata.

Observe que a FNBC é uma forma restritiva de 3FN, isto é, toda relação em FNBC está também em 3FN. Entretanto, uma relação em 3FN não está necessariamente em FNBC.

A *Quarta Forma Normal (4FN)* foi definida com base no conceito de dependência multivalorada, a qual pode ser entendida como uma generalização da 1FN. Outras formas normais foram definidas posteriormente, cuja aplicabilidade prática é duvidosa, já que se baseiam em conceitos difíceis de serem reproduzidos no mundo real.

8. Projeto físico de dados

Requisitos do projeto físico de dados

Projeto físico de banco de dados é o processo de escolher estruturas de armazenamento e caminhos de acesso específico para os arquivos do banco de dados, de modo a alcançar o melhor desempenho para as várias aplicações. Cada SGBD oferece opções para métodos de acesso e organização de arquivos, tais como: vários tipos de indexação, agrupamento de registros relacionados em blocos de discos, ligação de registros relacionados via ponteiros, indexação hashing, etc. O processo de design físico ficará restrito às opções oferecidas pelo SGBD escolhido. Outro aspecto importante é que, embora padronizados pela SQL-92, os tipos de dados variam consideravelmente de um SGBD para outro. Nesta fase de projeto, o projetista deve escolher os tipos de dados e formatos que melhor se adequem às especificações.

Após implementado, o sistema precisa ser ajustado com base em observações obtidas por monitoração. Os principais aspectos monitorados são os seguintes:

- *Tempo de resposta*: tempo decorrido entre a submissão da transação para execução e o recebimento da resposta;
- *Utilização de espaço*: quantidade de espaço de armazenamento utilizado pelas estruturas dos arquivos do banco de dados e seus métodos de acesso;
- *Throughput de transações*: número médio de transações que pode ser processado, no pico do uso, pelo sistema de banco de dados.

A informação sobre a frequência esperada de taxas de invocação, junto com a informação sobre campos coletada sobre cada consulta e transação, é usada para compilar uma lista cumulativa de frequência esperada de uso de todas as consultas e transações. Em situações práticas, vale a regra do “80-20”: basta coletar informação sobre os 20% mais importantes de consultas e transações que isto representará cerca de 80% do processamento.

Outra informação valiosa é a frequência esperada de operações de atualização. Se uma tabela vai ser atualizada com muita frequência, um número pequeno de caminhos de acesso deve ser especificado para a tabela porque a atualização das estruturas auxiliares tornará mais lenta as operações de atualização.

Os seguintes requisitos dependentes da tecnologia devem ser observados, devendo nortear o projeto físico de qualquer banco de dados: performance, economia, disponibilidade e segurança.

Performance

Um banco de dados é eficiente se ele propiciar um tempo de resposta satisfatório para as aplicações e usuários que o acessam.



Economia

Um banco de dados é “econômico” se ele utilizar o mínimo possível dos recursos computacionais disponíveis (espaço em disco, tempo de CPU, etc.). A economia tende a ser inversa em relação a PERFORMANCE.

Disponibilidade

Um banco de dados é dito disponível se ele estiver pronto para utilização o maior período de tempo possível. A disponibilidade depende dos seguintes fatores: mecanismo de controle de concorrência e mecanismo de controle de recuperação de falhas implementados pelo SGBD. Um sistema é dito disponível 99,9% das vezes se o somatório dos tempos de parada em um ano for de no máximo 8 horas.

Segurança

Um banco de dados é seguro se ele permitir que determinadas classes de objetos de dados sejam acessadas apenas por determinadas classes de usuários autorizadas para tal. A segurança depende diretamente do mecanismo de controle de segurança implementado pelo SGBD.