Banco de

Sumário

Prefácio

Esta apostila tem como objetivo introduzir o aluno no “mundo” dos Bancos de Dados. A maior parte do texto existente nesta apostila não é de nossa autoria, na verdade, o texto aqui contido é uma síntese de vários autores renomeados na área de Banco de Dados.

Porém, é de grande importância destacarmos, que esta apostila não substitui o valor presente nos livros de autores consagrados, ela é uma ferramenta para o aluno iniciar seus estudos.

Por causa da grande quantidade de trechos extraídos de alguns livros, fica impraticável referenciar todos eles, porém, todos os livros utilizados para compor esta apostila estão citados na bibliografia.

Lição 01
Conceitos Básicos

"Só sei que nada sei"

Sócrates

1. Objetivo

Como primeira lição, daremos aqueles conceitos necessários para se entender Banco de Dados. Mostraremos a importância dos Bancos de Dados e para aqueles mais leigos explicaremos o que é um Banco de Dados. Ao final desta lição esperamos ter o domínio dos conceitos básicos de Banco de Dados, tais como, abstração, instância, esquema, estrutura, modelo de dados, entre outros.

1.1. O que é Banco de Dados?

Se você não fez já deve ter pensado em fazer esta pergunta: *"O que é Banco de Dados?"*, poderíamos te responder: *"É um conjunto de dados"*. Note que o conceito é relativamente simples, mas a importância de um Banco de Dados é enorme, agora cada vez mais, já que estamos na *"era da informação"*.

Todo mundo possui alguma informação que deseja guardar, seja esta informação relativa a clientes, a produtos, a eleitores ou simplesmente a telefones de amigos, um conjunto de informação que possui alguma relação entre si formará o que chamamos de Banco de Dados.

Num computador o conjunto de dados é associado a um conjunto de programas para acessar estes dados, chamamos este sistema de *Sistema Gerenciador de Banco de Dados* (SGBD), note que o SGBD não é apenas o conjunto de dados, nem apenas o conjunto de programas, ele é os dois. O principal objetivo de um SGBD é proporcionar um ambiente tanto conveniente quanto eficiente para a recuperação e armazenamento das informações.

1.2.Objetivos dos Sistemas de Banco de Dados

Um modo, que já foi muito utilizado, de guardar as informações no computador foi armazená-las em sistemas de arquivos permanentes. Estes sistemas eram criados para satisfazer as necessidades "atuais" de uma empresa, com o passar do tempo eram adicionados novos módulos sobre os já existentes. Muitas vezes os novos módulos eram escritos por outros programadores, que por sua vez não utilizavam as mesmas linguagens. Com todas estas divergências aconteciam diversos problemas, que variavam desde redundância até isolamento de dados.

O Sistema de Banco de Dados veio evitar estes problemas. Os principais objetivos de um Sistema de Banco de Dados são:

- **Gerenciar grande quantidade de informação**: um Sistema de Banco de Dados pode armazenar simplesmente dados referentes a uma agenda de amigos, como também pode armazenar as informações relativas a uma usina nuclear. Em ambos os casos o Sistema de Banco de Dados tem que nos dar segurança e confiabilidade, independente se ele guardará 10 Megabytes ou 900 Gigabytes de informação.
- **Evitar redundância de dados e inconsistência**: redundância é manter a mesma informação em lugares diferentes, isto acontecia muito nos Sistemas de Arquivos, visto que novos programadores poderiam criar novos arquivos que conteriam um determinado dado que já está sendo armazenado em outro arquivo. Um dos problemas da redundância é que podemos atualizar um determinado dado de um arquivo e esta atualização não ser feita em todo o sistema, este problema é chamado de inconsistência. Um Sistema de Banco de Dados tenta evitar ao máximo estes erros, vendo ainda que a redundância causa desperdício de memória secundária e tempo computacional.
- **Facilitar o acesso**: um Sistema de Banco de Dados facilita ao máximo o acesso aos dados, vistos que estes dados estarão no mesmo formato, o mesmo não acontecia no Sistema de Arquivos, onde os dados poderiam estar em diversos formatos e o acesso poderia até ser impossível. Outro ponto que um Sistema de Banco de Dados facilita é o acesso concorrente, onde podemos ter a mesma informação sendo compartilhada por diversos usuários.

- **Segurança aos Dados**: nem todos os usuários de banco de dados estão autorizados ao acesso a todos os dados. Imagine, se numa empresa todos funcionários tivessem acesso à folha de pagamento. O Sistema de Banco de Dados garante a segurança implementando senhas de acessos.
- **Garantir a Integridade**: é fazer com que os valores dos dados atribuídos e armazenados em um banco de dados devam satisfazer certas restrições para manutenção da *consistência* e *coerência*. Por exemplo, não podemos permitir a entrada de números onde é para entrar a sigla do Estado.
- **Facilitar Migração se necessário**: às vezes por motivos de velocidade ou de atualização precisamos mudar todo o Sistema Computacional, e os dados serão armazenados em um outro Banco de Dados. O ato de Transferir as informações de um Banco de Dados para outro Banco de Dados é chamado de Migração, e facilitar esta Migração é um dos objetivos de um Sistema de Banco de Dados.

1.3.Componentes de um Sistema de Banco de Dados

Podemos dizer que um Sistema de Banco de Dados envolve 4 componentes básicos: Dados, Software, Hardware e Usuário.

Sobre o **Dado**, já falamos de maneira indireta, que é o componente principal, ele é as informações que serão armazenadas.

Hardware é toda a parte física, a máquina em si. Alguns o resumem apenas ao computador, mas é um erro, visto que mesmo um celular pode enviar e receber dados. Um ponto que sempre temos que destacar é que o *Hardware deve se adaptar ao BD* e não o contrário.

Software é toda a "parte lógica", os programas aplicativos, os programas de acesso aos dados, até mesmo o sistema operacional.

Sobre os **Usuários**, alguns livros os dividiram em três tipos: o Administrador de Banco de Dados, o Programador de Aplicativos e o Usuário Final. Apesar desta divisão esta totalmente correta, é mais interessante fazer uma divisão mais minuciosa, sendo cada tipo, diferenciado por suas expectativa de interação com o sistema. Assim podemos dizer que existem cinco tipos:

➢ *Programadores de Aplicações*: são profissionais em computação que interagem com o sistema por meio de chamadas DML (veremos mais à frente seu significado), as quais são envolvidas por programas escritos na linguagem hospedeira (por exemplo, C ou Delphi). Este usuário cria programas que acessam de alguma forma a base de dados.

Uma vez que a sintaxe da DML é, em geral, completamente diferente da sintaxe da linguagem hospedeira, as chamadas DML são, normalmente, precedidas por um caractere especial antes que o código apropriado possa ser gerado. Um pré-processamento, chamado pré-compilador DML, converte os comandos DML para as chamadas normais em procedimentos da linguagem hospedeira. O programa resultante é, então, submetido ao compilador da linguagem hospedeira, a qual gera o código de objeto apropriado.

➢ *Usuários sofisticados*: são aqueles que fazem solicitações ao Banco de Dados, sem fazer programas. Estas solicitações são feitas através de linguagens de consultas. Um exemplo deste tipo de usuário são os analistas que submetem consultas para explorar dados no Banco de Dados.

➢ *Usuários especialistas*: são usuários sofisticados que escrevem aplicações especializadas de Banco de Dados que não podem ser classificadas como aplicações tradicionais em processamento de dados. Estas aplicações incluem desde áudio até modelagem de ambientes.

➢ *Usuários navegantes*: também chamados de usuários ingênuos, são aqueles que interagem com o Banco de Dados utilizando um dos programas aplicativos.

➢ *Administrador de Banco de Dados (ABD ou DBA)*: é o responsável por todo o Banco de Dados, entre suas funções podemos destacar:

- Definição do Esquema: O ABD cria o esquema do banco de dados original escrevendo um conjunto de definições que são transformadas pelo compilador DDL em um conjunto de tabelas armazenadas de modo permanente no dicionário de dados.
- Definição da estrutura de dados e método de acesso: apropriados são escritos pelo ABD através de um conjunto de definições, as quais são traduzidas pelo compilador de armazenamento de dados e pelo compilador de linguagem de definição de dados.

- Esquema e modificações na organização física: os programadores realizam poucas alterações no esquema do Banco de Dados ou na descrição da organização física
- Fornecer autorização de acesso ao sistema: o ABD fornece a cada usuário um acesso restrito, permitindo melhor controle.
- Especificação de regras de integridade: o ABD específica restrições para manutenção de integridade dos dados.

1.4. Abstração de Dados

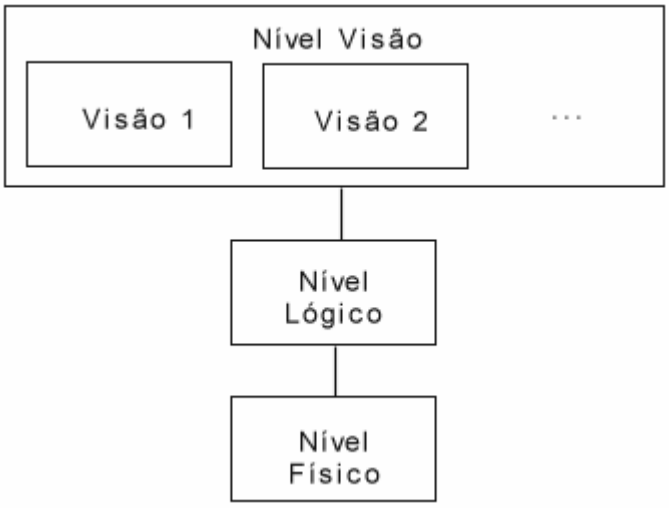
Num sistema de Banco de Dados o usuário vê uma visão abstrata, o sistema esconde certos detalhes de como os dados são armazenados ou mantidos. Isto porque, a preocupação com a eficiência de um Banco de Dados, leva a concepção de estruturas de dados complexas para a representação dos dados no Banco de Dados. Porém, uma vez que o Sistema de Banco de Dados são freqüentemente usados por leigos na área de algoritmo da computação, esta complexidade precisa ser escondida dos usuários; conseguimos isto com diferentes níveis de abstração: Nível Físico, Nível Conceitual e Nível Visão.

Nível Físico: é o nível mais baixo de abstração, no qual se descreve como os dados são realmente armazenados no "disco". Neste Nível estruturas complexas, de baixo nível, são descritas em detalhes.

Nível Conceitual: também chamado de *Nível Lógico*, é considerado o nível secundário de abstração, no qual se descreve quais dados são realmente armazenados no BD e quais os relacionamentos entre eles.

Nível Visão: o mais alto nível de abstração, no qual se expõem só a parte do Banco de Dados que um determinado usuário necessita. O sistema pode proporcionar diversas visões do mesmo Banco de Dados.

O inter-relacionamento entre esses três níveis de abstração está ilustrado na próxima figura:



1.5.Esquema

Esquema é a estrutura de um Banco de Dados. O Sistema de Banco de Dados apresentam diversos esquemas, referentes aos níveis de abstração que discutimos. No nível mais baixo há o esquema físico; no nível intermediário, o esquema lógico; e no nível mais alto, os sub-esquemas.

1.6.Instância

São os valores contidos no Banco de Dados em um determinado momento, isto é, o conjunto de informações contidas num Banco de Dados em um determinado instante.

1.7.Independência de Dados

A habilidade de modificar a definição de um esquema em um nível sem afetar a definição de esquema num nível mais alto é chamado de independência de dados. Existem dois níveis de independência de dados:

- **Independência física de dados** é a habilidade de modificar o esquema físico sem a necessidade de reescrever os programas aplicativos. As modificações no nível físico são ocasionalmente necessárias para aprimorar o desempenho.
- **Independência lógica de dados** é a habilidade de modificar o esquema conceitual sem a necessidade de reescrever os programas aplicativos. As

modificações no nível conceitual são necessárias quando a estrutura lógica do banco de dados é alterada (por exemplo, a adição de contas de bolsas de mercado num sistema bancário).

A independência lógica de dados é mais difícil de ser alcançada do que a independência física, pois os programas são bastante dependentes da estrutura lógica dos dados que eles acessam.

O conceito de independência de dados é similar em muitos aspectos ao conceito de tipos abstratos de dados em modernas linguagens de programação. Ambos escondem detalhes de implementação do usuário. Isto permite ao usuário concentrar-se na estrutura geral em vez de em detalhes de baixo nível de implementação.

1.8. Modelo de Dados

De forma a descrever a estrutura de um Banco de Dados, necessitamos definir o conceito de modelo de dados. Modelo de dados é uma coleção de ferramentas conceituais para a descrição dos dados, relacionamento entre os dados e restrições dos dados. Diversos modelos de dados foram propostos e estão divididos em três diferentes grupos:

- modelos lógicos baseados em objetos;
- modelos lógicos baseados em registro;
- modelos físico.

1.8.1. Modelos lógicos baseados em Objetos

Usados na descrição de dados nos níveis conceitual e visão, proporcionam ampla e flexível capacidade de estruturação e permitem a especificação de restrições de dados de forma explicita. Na última contagem, levantamos pelo menos 30 modelos diferentes e mais modelos logo surgirão. Entre os modelos mais conhecidos estão:

- Modelo Entidade-Relacionamento;
- Modelo orientado a objeto;
- Modelo semântica de dados;
- Modelo funcional de dados.

O modelo Entidade-Relacionamento esta sendo largamente utilizado na prática, ele se baseia numa percepção do mundo real e consiste numa coleção de objetos básicos chamados de *entidades* e de *relacionamento* entre estes objetos.

1.8.2. Modelos lógicos baseados em Registros

São usados na descrição de dados nos níveis conceitual e visão, especificam tanto a estrutura global, como uma descrição em auto-nível da implementação, dividem-se em: Modelo Hierárquico, Modelo Rede e Modelo Relacional.

Modelo Hierárquico: neste modelo os dados são representados por registros e os relacionamentos por ligações (ponteiros). Os dados são organizados em uma estrutura de árvore que se origina a partir de uma raiz, e se ramifica em nós (coleção de atributos que descreve uma entidade ou registro). Os relacionamentos são de pai-para-filho, o nó que não possui filho é chamado de folha.

Vantagem do Modelo Hierárquico: implementação simples e eficiente, maturidade, confiança.

Desvantagem do Modelo Hierárquico: não se consegue modelar estruturas que são de M-para-M, ao excluir um elemento temos que remontar toda estrutura do banco.

Modelo Rede: apesar dos conceitos que existem por trás do modelo Rede serem originários dos anos 60. As primeiras especificações escritas foram feitas em 1971 pela Conference on Data System Languagens (codasyl), por este motivo, é comum vermos o Modelo Rede ser chamado de Modelo Codasyl. Este modelo descreve os Banco de Dados nos quais existem relações de M-para-M, as ligações entre os registros são ponteiros.

Vantagem do Modelo Rede: seu triunfo é permitir relacionamentos de M-para-M, além de não precisarmos alterar toda a estrutura numa exclusão ou inclusão. Desvantagem do Modelo Rede: o mapeamento pode se tornar complexo.

Modelo Relacional: os dados e os relacionamentos entre os dados são representados por uma coleção de tabelas, cada qual com múltiplas colunas e nome único.

Vantagem do Modelo Relacional: simplicidade do uso, o relacionamento é feito por campo “chave” e independência dos dados.

Desvantagem do Modelo Relacional: desempenho ainda esta em evolução.

1.8.3. Modelos Físicos

São aqueles que se preocupam com a descrição dos dados no nível mais baixo, se preocupam com detalhes de como são armazenados os dados. Ao contrário dos modelos lógicos, há poucos modelos físicos em uso. Dois deles são amplamente conhecidos: o modelo unificado (unifying model) e o modelo de partição de memória (frame memory model).

1.9.Linguagens de Banco de Dados

Um sistema de Banco de Dados proporciona dois tipos de linguagens: uma específica para os esquemas do Banco de Dados e outra para expressar consultas e atualizações.

1.9.1. Linguagens de Definição de Dados

Um esquema de dados é especificado por um conjunto de definições expressas por uma linguagem especial chamada linguagem de definição de dados (data-definition language – DDL). O resultado da compilação dos parâmetros DDLs é armazenado em um conjunto de tabelas que constituem um arquivo especial chamado dicionário de dados ou diretório de dados.

Um dicionário de dados é um arquivo de metadados – isto é, dados a respeito de dados. Em um sistema de banco de dados, esse arquivo ou diretório é consultado antes que o dado real seja modificado.

A estrutura de memória e o método de acesso usados pelo banco de dados são especificados por um conjunto de definições em um tipo especial de DDL chamado linguagem de definição e armazenamento de dados (data storage and definition language). O resultado da compilação dessas definições é um conjunto de instruções para especificar os detalhes de implementação dos esquemas do banco de dados – os detalhes normalmente são ocultados dos usuários.

1.9.2. Linguagens de Manipulação dos Dados

Os níveis de abstração que foram discutidos não se aplicam apenas à definição ou à estrutura dos dados, mas também a sua manipulação.

Por manipulação de dados entendemos:

- A recuperação das informações armazenadas no banco de dados;
- Inserção de novas informações no banco de dados;
- A remoção de informações do banco de dados;
- A modificação das informações do banco de dados.

A linguagem de manipulação de dados (DML) é a linguagem que viabiliza o acesso ou a manipulação dos dados de forma compatível ao modelo de dados apropriado. São basicamente dois tipos:

- *DMLs procedurais*: exigem que o usuário especifique quais dados são necessários, e como obtê-los.
- *DMLs não procedurais*: exigem que o usuário especifique quais dados são necessário sem especificar como obtê-los.

Uma consulta é uma solicitação para recuperação de informações. A parte de uma DML responsável pela recuperação de informações é chamada linguagem de consultas (query language). Embora tecnicamente incorreto, é comum o uso do termo linguagem de consultas como sinônimo de linguagem de manipulação de dados.

1.10. Gerenciamento de Transações

Freqüentemente, muitas operações em um Banco de Dados constituem uma única unidade lógica de trabalho. Uma transação é uma coleção de operações que desempenham uma função lógica única dentro de uma aplicação do sistema de Banco de Dados. Cada transação é uma unidade de atomicidade (ou tudo ou nada) e consistência (exigência de corretismo). Assim o Banco de Dados permanece consistente após a transação, mas durante ela será necessário aceitar inconsistências temporárias, que podem gerar problemas caso ocorra uma falha.

É responsabilidade do programador definir, de modo apropriado, as diversas transações, tais que cada uma preserve a consistência do Banco de Dados. Caso ocorra alguma falha na transação, o Banco de Dados terá que assumir a última forma consistente dele.

1.11. Administração de Memória

Normalmente, os Bancos de Dados exigem um grande volume de memória. Um banco de Dados corporativo é usualmente medido em *gigabytes* ou para Banco de Dados de grande porte (largest database), *terabytes*. Isto tudo de informação não pode

ficar na memória principal, necessitando ser armazenada em “disco”. Então, os dados são transferidos do “disco” para a memória principal quando necessário. Esta transferência é muito lenta, fazendo com que seja imperativo estruturar o Banco de Dados para minimizar este movimento.

Um gerenciamento de memória é um módulo de programas para interface entre o armazenamento de dados em um nível baixo e consultas e programas de aplicação submetidos ao sistema. O gerenciamento de memória é responsável pela interação com o gerenciamento de arquivos. O gerenciador de memória traduz os diversos comandos DML em comandos de baixo nível de sistema de arquivos. Assim, o gerenciador de memória é responsável pelo armazenamento, recuperação e atualização de dados no banco de dados.

1.12. Estrutura Geral do Sistema

Um sistema de bancos de dados é dividido em módulos que tratam de cada uma das responsabilidades do sistema geral. Na maioria dos casos, o sistema operacional do computador fornece apenas os serviços mais básicos, e o sistema de bancos de dados precisa ser construído sobre essa base. Então, o projeto do sistema de bancos de dados precisa incluir considerações sobre a interface entre o sistema de bancos de dados e o sistema operacional.

Os componentes funcionais de um sistema de bancos de dados podem ser divididos pelos componentes de processamento de consultas e pelos componentes de administração de memória. Os componentes de processamento de consultas incluem:

- **Compilador DML**, que traduz comandos DML da linguagem de consulta em instruções de baixo nível, inteligíveis ao componente de execução de consultas. Além disso, o compilador DML tenta transformar a solicitação do usuário em uma solicitação equivalente, mas mais eficiente, buscando, assim, uma boa estratégia para a execução da consulta.
- **Pré-compilador DML** que converte comandos DML embutidos em um aplicativo para chamadas de procedimento normal na linguagem hospedeira. O pré-compilador precisa interagir com o processador de consultas para gerar o código apropriado.

- **Interpretador DDL** que interpreta os comandos DDL em um conjunto de tabelas contendo metadados, ou seja, “dados sobre dados”.
- **Componentes para o tratamento de consultas**, que executam instruções de baixo nível geradas pelo compilador DML.

Os componentes para administração do armazenamento de dados proporcionam a interface entre os dados de baixo nível, armazenados no Banco de Dados, os programas de aplicações e as consultas submetidas ao sistema. Os componentes de administração de armazenamento de dados incluem:

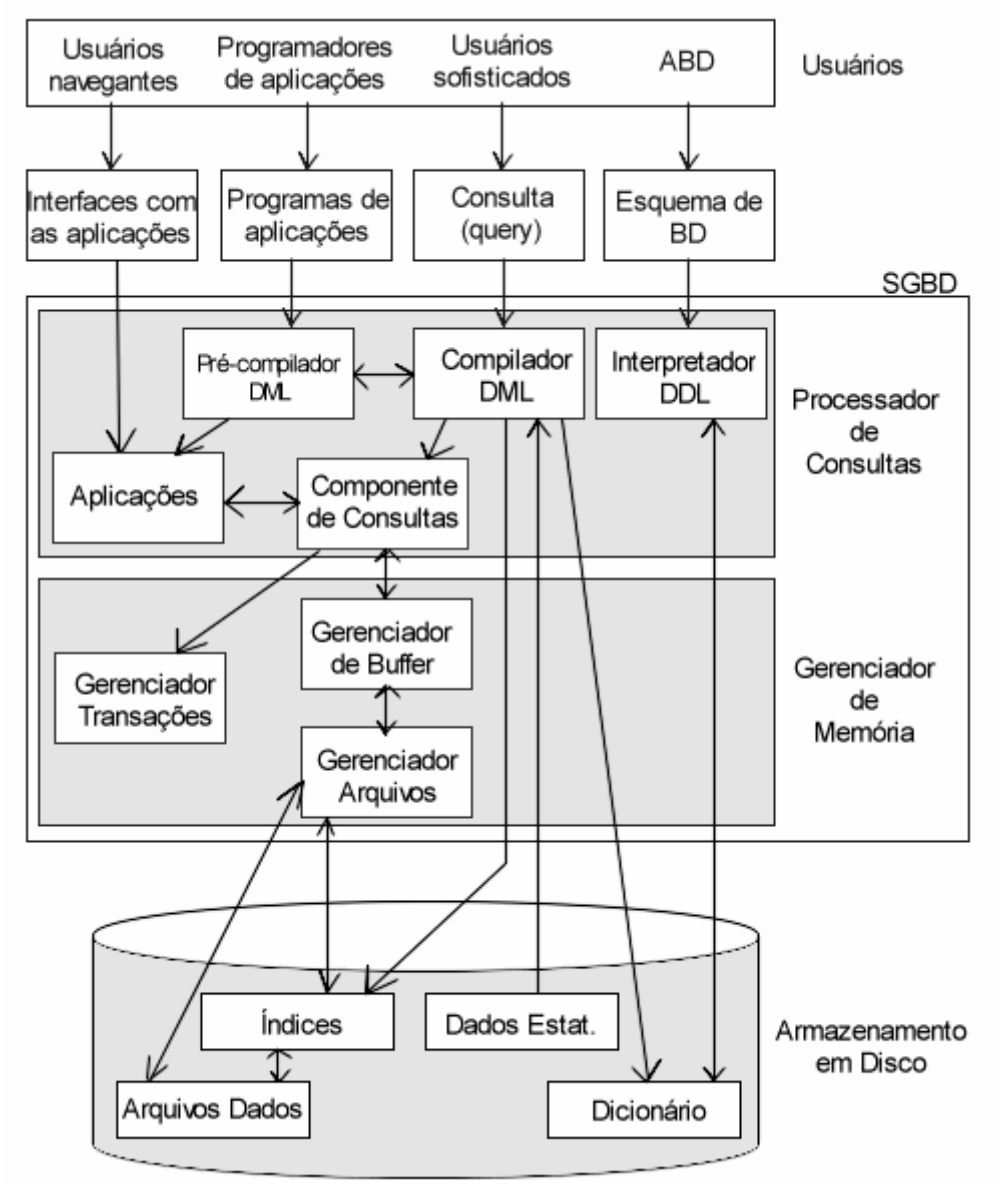
- **Gerenciamento de autorizações e integridade**, que testam o cumprimento das regras de integridade e a permissão ao usuário no acesso ao dado.
- **Gerenciamento de transações**, que garante que o Banco de Dados permanecerá em estado consistente a despeito de falhas no sistema e que transações concorrentes serão executadas sem conflitos em seus procedimentos.
- **Administrador de arquivos**, que gerencia a alocação do espaço na armazenagem do disco e as estruturas de dados usadas para representar a informação armazenada no disco.
- **Administração de buffer**, responsável pela intermediação de dados do disco para a memória principal e pela decisão de quais dados colocar em memória cache.

Adicionalmente, diversas estruturas de dados são requisitadas como parte da implementação do sistema físico, incluindo:

- **Arquivos de dados** que armazenam banco de dados por si mesmos.
- **Dicionário de dados** que armazenam metadados sobre a estrutura do banco de dados. O dicionário de dados é bastante usado. Assim, deve-se dar grande ênfase ao desenvolvimento de um bom projeto e eficiente implementação ao dicionário.
- **Índices** que fornecem acesso rápido aos itens de dados guardando dados particulares.
- **Estatísticas de dados**, armazenam informações estatísticas relativas aos dados contidos no Banco de Dados. Essas informações são usadas pelo

processador de consultas para seleção de meios eficientes para execução de uma consulta.

A figura na próxima página representa, de forma resumida, todo o sistema de Banco de Dados.



Lição 02
Modelo Relacional

"Penso, logo existo!"

Descartes

2. Objetivo

O Modelo Relacional, em nível comercial, esta em primeiro lugar entre os modelos de dados. Esta Lição tem como objetivo mostrar a estrutura dos Bancos de Dados Relacionais, bem como, introduzir o aluno aos conceitos e teorias relacionadas ao Modelo Relacional.

2.1. Estrutura dos Bancos de Dados Relacionais

Um Banco de Dados Relacional consiste em uma coleção de *tabelas*, cada uma das quais com um nome único. Representamos em tabelas o banco de Dados E-R (que veremos mais a frente). Uma linha em uma tabela representa um *relacionamento* entre um conjunto de valores. Uma vez que essa tabela é uma coleção de tais relacionamentos, há uma estreita correspondência entre o conceito de tabela e o conceito matemático de relação, a partir das quais se origina o nome desse modelo de dados.

2.2.1. Estrutura Básica

Observe a tabela *cliente*:

Código	Nome	Cidade
011	Ana Fernanda	Sorocaba
012	Joseph	Laguna
013	Marcos Túlio	Capivari de Baixo
014	Mariana	Blumenau

Ela possui três colunas: *Código*, *Nome* e *Cidade*. Os nomes das colunas são chamados de atributos. Para cada atributo, há um conjunto de valores permitidos, chamado *domínio* do atributo em questão. Para o atributo *Cidade*, por exemplo, o domínio é o conjunto de todos os nomes de cidades. Suponha que D_3 denote esse

conjunto, D_2 denota o conjunto de todos os nomes de clientes e D_1, o conjunto de todos os códigos. Qualquer linha de *cliente* consiste necessariamente de uma 3-tupla (v_1,v_2,v_3), em que v1 é um código de cliente (isto é, v_1 está no domínio D_1), v_2 é um nome de cliente (isto é, v_2 está no domínio D_2) e v_3 é uma cidade (isto é, v_3 está no domínio D_3). Em geral, a tabela *cliente* é um subconjunto de:

$$D_1 \times D_2 \times D_3$$

Em geral, uma tabela de *n* atributos deve ser um subconjunto de:

$$D_1 \times D_2 \times ... \times D_{n-1} \times D_n$$

Matematicamente, define-se uma relação como um subconjunto de um produto cartesiano de uma lista de domínios. Essa relação corresponde quase exatamente à definição de uma tabela. A única diferença é que designamos nomes aos atributos, ao passo que, matematicamente, se usam apenas "nomes" numéricos. Podemos então, utilizar os termos matemáticos *relação* e *tupla* no lugar de tabela e linhas.

Na relação *cliente* existem quatro tuplas. Suponha que a *tupla variável t* se refira à primeira tupla da relação. Usamos a notação t[código] para denotar o valor de t no atributo código. Assim, t[código]=011 e t[nome]="Ana Fernanda". Alternativamente podemos escrever t[1] para denotar os valores da tupla t no primeiro atributo, t[2] para denotar nome e assim por diante. Já que uma relação é um conjunto de tuplas podemos usar a notação matemática $t \in r$ para denotar que a tupla t está na relação r.

Exigimos que, para todas as relações r, os domínios de todos os atributos de r sejam atômicos. Um domínio é atômico se elementos desse domínio são considerados unidades indivisíveis.

Um valor de domínio que pertença a qualquer domínio possível é o valor *nulo*, que indica que um valor é desconhecido ou não existe.

2.2.2. Esquema de Banco de Dados

O conceito de *esquema de relação* corresponde, em linguagem de programação, à noção de definição de tipos. É conveniente dar um nome ao esquema de relação, assim como damos nomes aos tipos em linguagem de programação. Adotamos convencionar o uso de letras minúsculas para nomes de relações e nomes iniciados com

uma letra maiúscula para esquemas de relações. Seguindo essa notação, usamos o nome Esquema_cliente para denotar o esquema de relação para a relação cliente. Assim,

Esquema_cliente = (código, nome, cidade)

Denotamos o fato de cliente ser uma relação em Esquema_cliente por:

cliente(Esquema_cliente)

O conceito de *instância de relação* corresponde, em linguagem de programação, ao valor de uma variável. O conteúdo de uma instância pode mudar ao longo do tempo, quando esta relação é atualizada. Muitas vezes, usamos simplesmente "relação", quando na realidade nos referimos à "instância de relação".

2.2.3. Chaves

As noções de chave primária, chave candidata, chave secundária, que serão discutidos na lição sobre *Modelagem de Dados*, se aplicam também ao modelo relacional. Por enquanto é importante sabermos que, se um esquema de banco de dados relacional tem por base tabelas derivadas de um esquema E-R, é possível determinar a chave primária para um esquema de relação das chaves primárias dos conjuntos de relacionamentos ou entidades dos quais os esquemas são derivados.

2.2.4. Linguagens de Consulta

Uma *linguagem de consulta* é a linguagem por meio da qual os usuários obtêm informações do banco de dados. Os sistemas de banco de dados comerciais oferecem linguagem de consulta de alto nível, uma das mais populares é a SQL, veremos nesta lição linguagens "puras": a álgebra relacional, cálculo relacional de um tupla e o cálculo relacional de um domínio. Essas linguagens de consulta são concisas e formais, sem "o açúcar sintático" das linguagens comerciais, mas ilustram as técnicas fundamentais para a extração de dados do banco de dados.

2.3. A Álgebra Relacional

A álgebra relacional é uma linguagem de consultas procedural. Consiste em um conjunto de operações tendo como entrada uma ou duas relações e produzindo, como resultado, uma nova relação. As operações fundamentais na álgebra relacional são select, project, union, set, difference, cartesian product e rename. Além dessas operações fundamentais, existem algumas outras operações – namely, set intersection,

natural join, division e assignment. Essas operações são definidas em termos das operações fundamentais.

2.3.1. Operações Fundamentais

As operações select, project e rename são chamadas de operações primárias, pois operam uma única relação. As outras três relações operam um par de relações e são, portanto, chamadas de operações binárias.

2.3.1.1. A Operação Select

A operação *select* seleciona tuplas que satisfaçam um determinado predicado. Usamos a letra minúscula sigma (σ) para denotar seleção. O predicado aparece subscrito a σ. O argumento de relação é dado entre parênteses, seguindo o σ . Assim, para selecionar aquelas tuplas da relação cliente, cuja cidade é "Tubarão", escrevemos:

$$\sigma_{\text{cidade = "SOROCABA"}}(\text{cliente})$$

Podemos usar comparações do tipo =, ≠, <, ≤, > e ≥. Além de combinar vários predicados em um grande predicado usando os conectivos *e* (∧) e *ou* (∨). Por exemplo, queremos todos clientes de São Paulo que tenham código superior a 012:

$$\sigma_{\text{cidade = "São Paulo"} \ \wedge \ \text{código>012}} (\text{cliente})$$

A seleção do predicado pode ter comparação entre dois atributos.

2.3.1.2. A Operação Project

A operação *project* permite que listamos apenas um subconjunto dos atributos. Por exemplo, permite que façamos uma lista apenas com o nome e a cidade do cliente. Já que a relação é um conjunto, quaisquer linhas em duplicidade são eliminadas. A projeção é denotada pela letra grega pi(Π). Listamos, subscrito em Π, os atributos que desejamos na consulta. O argumento da relação vem entre parênteses, a seguir.

Para exemplificar, faremos uma projeção com os atributos nome e cidade:

$$\Pi_{\text{nome, cidade}} (\text{cliente})$$

A relação resultante dessa consulta é:

Nome	Cidade
Ana Fernanda	Tubarão
Joseph	Laguna
Marcos Túlio	Capivari de Baixo
Mariana	Blumenau

2.3.1.3. A Operação Relacional de Comparação

O fato de o resultado de uma operação relacional ser uma relação é importante, pois poderemos combinar as operações. Em vez de dar o nome da relação como argumento, podemos dar uma outra operação que evolui para uma relação. Por exemplo:

$$\Pi_{\text{nome}} (\sigma_{\text{cidade} = \text{“Capivari”}} (\text{cliente}))$$

Com a consulta acima, teremos uma lista com nomes de cliente que moram em Capivari. Em geral, desde que o resultado de uma operação em álgebra relacional seja do mesmo tipo que sua entrada(relação), as operações em álgebra relacional podem ser compostas juntas em uma expressão em álgebra relacional. A composição de operações em álgebra relacional em expressões é similar à composição de operações aritméticas em expressões aritméticas.

2.3.1.4. A Operação Union

A operação *union* é utilizado quando necessitamos uma relação que será gerada por relações que serão criadas por mais de uma operação. Por exemplo queremos os nomes de clientes e fornecedores:

$$\Pi_{\text{nome}}(\text{cliente}) \cup \Pi_{\text{nome}}(\text{fornecedor})$$

Em geral, precisamos que uniões sejam feitas entre relações compatíveis, sendo assim: as relações devem possuir o mesmo número de atributos e os domínios do i-ésimo atributo de uma relação e o i-ésimo atributo de outra devem ser os mesmos, para to i. Lembrando que as relações podem ser temporárias, resultados de uma expressão em álgebra relacional.

Outro ponto a destacar, é que como as relações são conjuntos, valores duplicados são eliminados.

2.3.1.5. A Operação Diferença entre Conjuntos

A operação *diferença entre conjuntos*, denotada por -, permite-nos encontrar as tuplas que estão em uma relação, mas não em outra. A expressão r - s resulta na relação que contém as tuplas que estão em r, mas não em s. Por exemplo, se quisermos os nomes dos clientes que não são fornecedores:

$$\Pi_{nome}(cliente) - \Pi_{nome}(fornecedor)$$

Como na operação de união, precisamos assegurar que o conjunto diferença seja feito entre relações compatíveis.

2.3.1.6. A Operação Produto Cartesiano

A operação *produto cartesiano*, representada por x, permite-nos combinar informações de duas relações quaisquer. Representamos o produto das relações r_1 e r_2 por r_1 x r_2. Observe a relação fornecedor:

Código	Nome	Cidade
001	João	Tubarão
002	Lucas	Laguna

A expressão: cliente x fornecedor, vai gerar a seguinte relação:

Código	Nome	Cidade	Código	Nome	Cidade
011	Ana Fernanda	Sorocaba	001	João	Sorocaba
011	Ana Fernanda	Sorocaba	002	Lucas	Laguna
012	Joseph	Laguna	001	João	Sorocaba
012	Joseph	Laguna	002	Lucas	Laguna
013	Marcos Túlio	Capivari de Baixo	001	João	Sorocaba
013	Marcos Túlio	Capivari de Baixo	002	Lucas	Laguna
014	Mariana	Blumenau	001	João	Sorocaba
014	Mariana	Blumenau	002	Lucas	Laguna

Podemos “misturar” as operações para gerar uma relação que precisarmos. Como pode existir atributos com mesmo nome, utilizamos para se referenciar a um atributo (deste tipo) a seguinte notação: nome da relação seguido por ponto(.) e por último o nome do atributo.

2.3.1.7. A Operação Rename

O resultado de uma expressão em álgebra relacional não possui um nome que possa ser usado para referenciá-la. O operador *rename*, representado pela letra

minúscula rho (ρ), permite executar esse tipo de tarefa. Dada a expressão em álgebra relacional E, a expressão:

$$\rho_x(E)$$

tem por resultado a expressão E sob o nome x.

Uma relação r é considerada como uma expressão (trivial) de álgebra relacional. Assim, podemos também aplicar a operação rename à relação r para obter a mesma relação sob um novo nome.

A segunda forma de usar a operação rename é a que se segue. Assuma que uma expressão E em álgebra relacional seja de ordem primária. Então a expressão:

$$\rho_{x(A1, A2, \ldots Na)}(E)$$

retorna o resultado da expressão E sob o nome x, com os atributos recebendo novos nomes, A1, A2, ..., An.

2.3.2. Definição Formal da Álgebra Relacional

Uma expressão básica na álgebra relacional pode ser uma das duas a seguir:

- Uma relação do banco de dados.
- Uma relação constante

Uma expressão genérica em álgebra relacional pode dar origem a sub-expressões menores. Seja E_1 e E_2 uma expressão em álgebra relacional. Então as expressões a seguir são todas expressões em álgebra relacional:

- $E_1 \cup E_2$
- $E_1 - E_2$
- $E_1 \times E_2$
- $\sigma_P(E_1)$, em que P é um predicado dos atributos em E_1.
- $\Pi_S(E_1)$, em que S é uma lista consistindo de alguns atributos em E_1.
- $\rho_x(E_1)$, em que x é um novo nome para o resultado de E_1.

2.3.3. Operações Adicionais

As operações adicionais na álgebra relacional são suficientes para expressar qualquer consulta em álgebra relacional. Entretanto, se nos restringirmos às operações fundamentais, certas consultas comuns tornam-se extensas. Portanto, definiremos operações adicionais que não conferem maior poder à álgebra relacional, mas

simplificam bastante consultas comuns. Para cada nova operação, daremos uma expressão equivalente usando somente operações fundamentais.

2.3.3.1. A Operações de Interseção de Conjuntos

A Operação de i*nterseção*, é representada pelo caracter ∩, ela retorna uma relação formada pelas tuplas que duas relações possuem em comum.

Suponhamos duas relações *r* e *s*, a expressão r ∩ s, retornará uma relação com apenas as tuplas que existem em r e também existem em s.

A operação fundamental de r ∩ s, seria r – (r - s).

2.3.3.2. A Operações de Junção Natural

A *junção natural* (natural join) é uma operação binária que nos permite combinar certas seleções e um produto cartesiano dentro de uma operação. As operações de junção natural formam um produto cartesiano de seus dois argumentos, promovem uma seleção obedecendo à equivalência dos atributos que aparecem em ambos os esquemas de relação e, finalmente, removem os atributos em duplicidade.

O caracter que representa a junção é |><| (“join). Uma junção pode ser escrita com operações fundamentais:

$$r \text{ |><| } s = \Pi_{R \cup S} (\sigma_{r.A1= s.A1 \wedge r.A2=s.A2 \wedge \ldots \wedge r.Na=s.An} \; r \times s)$$

2.3.3.3. A Operações de Divisão

A operação de *divisão*, simbolizada pelo símbolo ÷, é usada nas consultas nas quais se emprega a frase “para todos”. Suponha que tenhamos duas relações, a *r*:

nome_cliente	nome_agência
Jones	Downtown
Smith	Mianus
Hayes	Perryridge
Turner	Round Hill
Williams	Perryridge
Lindsay	Redwood
Johnson	Brighton
Jones	Brighton

e a *s*:

nome_agência
Brighton
Downtown

Desejamos encontrar todos os clientes que tenham conta em todas as agências: r ÷ s, o resultado para essa expressão é uma relação que possui o esquema (nome_cliente) e que contém a tupla (Jones).

Usando operações fundamentais:

$$r \div s = \Pi_{R-S}(r) - ((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$$

2.3.3.4. A Operações de Designação (Assignment Operation)

É conveniente, às vezes, escrever expressões em álgebra relacional com uma designação para a relação, de modo a usá-la como uma variável temporária. A operação de designação para a relação, denotada por ←, trabalha de maneira similar à designação (assignment) em linguagens de programação. Observe o exemplo:

$temp1 \leftarrow \Pi_{R-S}(r)$

$temp2 \leftarrow \Pi_{R-S}(r)((temp1 \times s) - r)$

resultado = temp1 – temp2

2.4. Cálculo Relacional de Tupla

O cálculo de uma tupla relacional é uma linguagem de consultas não-procedural. Ela permite a descrição da consulta desejada sem especificar os procedimentos para obtenção dessas informações.

Uma consulta em cálculo relacional de tupla é expressa como:

$$\{t \mid P(t)\}$$

que significa o conjunto de todas as tuplas t tal que o predicado P é verdadeiro para t. Seguindo nossa notação anterior, usamos t[A] para denotar os valores das tuplas t sobre os atributos A, e usamos $t \in r$ para denotar que a tupla t é uma relação r. Na definição temos que P é uma fórmula, algumas variáveis tuplas podem aparecer na fórmula.

2.5. Cálculo Relacional de Domínio

Existe uma Segunda forma de cálculo relacional chamada *cálculo relacional de domínio*. Essa forma usa variáveis de domínio que tomam valores do domínio de um

atributo, em vez de valores da tupla inteira. O cálculo relacional de domínio, entretanto, está intimamente relacionado ao cálculo da tupla relacional.

Uma expressão no cálculo relacional de domínio é da forma:

$$\{< x_1,x_2,...,x_n > | P(x_1,x_2,...,x_n)\}$$

em que $x_1,x_2,...,x_n$ representam variáveis de domínio e P representa uma fórmula tal e qual como no cálculo relacional de tuplas.

2.6. Operações de Álgebra Relacional Estendida

As operações básicas de álgebra relacional foram estendidas de diversas formas. Uma extensão simples é permitir operações aritméticas como parte da projeção. Uma extensão importante são as operações agregadas, como as que computam a soma de elementos de um conjunto ou sua média.

Outra importante extensão são as operações de junção externa (outer-join), as quais permitem expressões em álgebra relacional para o tratamento de valores nulos, como modelo de informações omitidas.

2.7. Modificações no Banco de Dados

Podemos consultar, adicionar, remover e alterar informações do Banco de Dados, até esse momento vimos somente a consulta. Expressamos as modificações no Banco de Dados usando as operações de designação.

2.7.1. Exclusão

A solicitação de uma exclusão é muitas vezes expressa do mesmo modo que uma consulta. Entretanto, em vez de mostrar as tuplas ao usuário, removemos as tuplas selecionadas do Banco de Dados. Podemos excluir somente tuplas inteiras, não podemos eliminar valores de um atributo em particular. Na álgebra relacional, a exclusão é expressa por:

$$r \leftarrow r - E$$

em que r é uma relação e E, uma consulta álgebra relacional.

2.7.2. Inserção

Para inserir dados em uma relação, podemos especificar uma tupla para inserção ou escrever uma consulta que resulte em um conjunto de tuplas a inserir. Obviamente, os valores dos atributos das tuplas a inserir devem ser membros do domínio dos atributos. Uma inserção é expressa por

$$r \leftarrow r \cup E$$

em que r é uma relação e E, uma expressão em álgebra relacional.

2.7.3. Atualização

Em certas situações, podemos desejar mudar o valor de uma tupla sem mudar todos os seus valores. Podemos usar um operador de projeção generalizada para essa tarefa, aqui não entraremos em detalhes, mais informações sobre projeção generalizada pode ser pesquisadas em livros que tratam de Álgebra Relacional Estendida.

2.8. Visões

Até agora usamos operadores no nível lógico. Isto é, assumimos que a coleção de relações dadas sejam, na verdade, relações armazenadas no Banco de Dados. Muitas vezes porém, por questão de segurança, necessitamos criar visões e deixar que o usuário só trabalhe com estas visões.

Para se definir uma visão usamos o comando create view. Para definir uma visão, precisamos dar um nome a ela e definir a consulta que criará essa visão. A forma do comando create view é:

create view v **as** <expressão de consulta>

onde v é o nome da visão, <expressão de consulta> é a expressa que criará a visão.

Uma vez definida a visão, podemos usar o nome da visão para nos referirmos à relação virtual gerada por essa visão.

Embora visões sejam ferramentas úteis para consultas, criam problemas significativos se atualizações, inserções ou exclusões são expressas sobre elas. A dificuldade é que uma modificação no Banco de Dados, expressa em termos de visão,

deverá se transformar em uma modificação nas relações reais do modelo lógico do Banco de Dados.

Um ponto a destacar, é que a partir de visões podemos criar outra visões.

Lição 03
Arquiteturas de SBD

"Saber é poder!"

Bacon

3. Objetivo

Nesta lição, o aluno terá uma descrição de algumas Arquiteturas de Banco de Dados, além é claro, de receber noções do que é Arquitetura de Banco de Dados e o que a influenciam. Esta lição é essencial para podermos compreender melhor as lições 04 e 05.

3.1. Arquiteturas

A arquitetura de um Sistema de Banco de Dados é fortemente influenciada pelo sistema básico computacional sobre o qual o sistema de dados é executado. Aspectos da arquitetura de computadores – como rede, paralelismo e distribuição – têm influência na arquitetura do Banco de Dados.

- Rede de computadores permite que algumas tarefas sejam executadas no servidor do sistema e outras sejam executadas no cliente. Essa divisão de trabalho tem levado ao desenvolvimento de sistemas de dados *cliente-servidor*.
- Processamento paralelo em um sistema de computadores permite que atividades do Sistema de Banco de Dados sejam realizadas com mais rapidez, reduzindo o tempo de resposta das transações e, assim, aumentando o número de transações processadas por segundo. Consultas podem ser processadas de forma a explorar o paralelismo oferecido pelo sistema operacional. A necessidade de processamento paralelo de

consultas tem levado ao desenvolvimento de sistemas de banco de dados paralelos.

- A distribuição de dados pelos nós da rede ou pelos diversos departamentos de uma organização permitem que esses dados residam onde são gerados ou mais utilizados, mas, ainda assim, estejam acessíveis para outros nós de outros departamentos. Dispor de diversas cópias de um Banco de Dados em diferentes nós também permite a organizações de grande porte manter operações em seus Bancos de Dados mesmo quando um nó é afetado por um desastre natural, como inundações, incêndios ou terremotos. Sistemas de Banco de Dados distribuídos têm se desenvolvido para tratar dados distribuídos geográfica ou administrativamente por diversos sistemas de Banco de Dados.

Estudaremos a arquitetura dos sistemas de banco de dados, começando com os sistemas centralizados tradicionais e passando por sistemas de Banco de Dados cliente-servidor (mais detalhes na lição 04), paralelos e distribuídos (mais detalhes na lição 05).

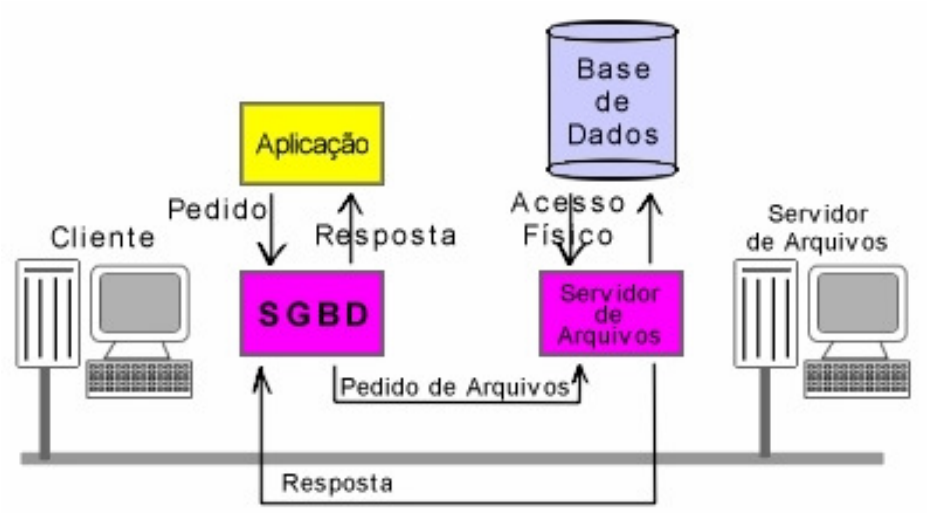
3.2. Sistemas Centralizados

Sistemas de banco de dados centralizados são aqueles executados sobre um único sistema computacional que não interagem com outros sistemas. Tais sistemas podem ter a envergadura de um sistema de Banco de Dados de um só usuário executado em um computador pessoal até sistemas de alto desempenho em sistemas de grande porte.

Um sistema computacional genérico moderno consiste em uma ou poucas CPUs e dispositivos de controle que são conectados por meio de um *bus* comum que proporciona acesso à memória compartilhada. As CPUs têm memórias *cache* locais que armazenam cópias de partes da memória para acesso rápido aos dados. Cada dispositivo de controle atende a um tipo específico de dispositivo. A CPU e os dispositivos de controle podem trabalhar concorrentemente, competindo pelo acesso à memória. A memória cache parcialmente reduz a contenção de acesso à memória, uma vez que reduz o número de tentativas de acesso da CPU à memória compartilhada.

3.3. Sistemas Cliente/Servidor

As aplicações baseadas no acesso a banco de dados podem ser implementadas de duas maneiras. Na primeira, utilizamos um sistema gerenciador de bancos de dados (SGBD) executando no cliente, que usará um servidor de arquivos para armazenar os arquivos do banco de dados, observe:

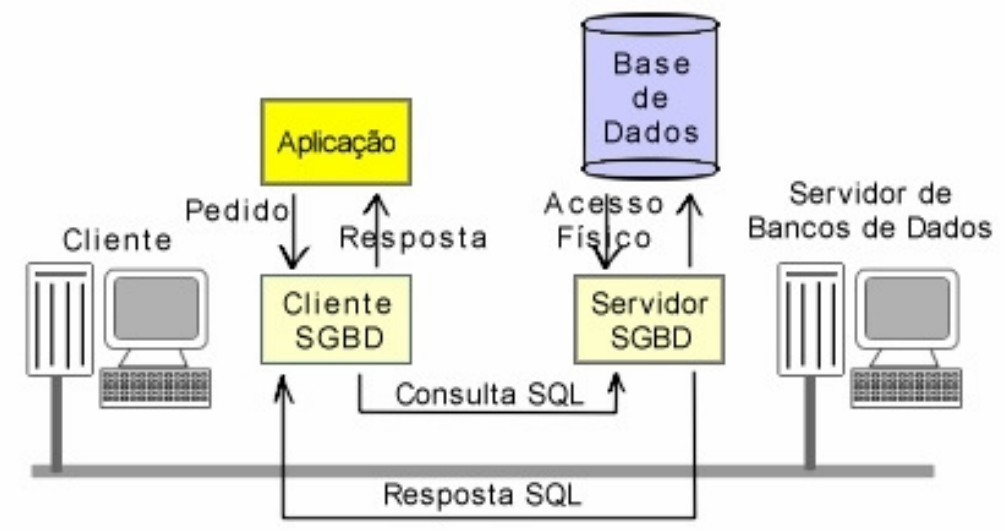


Essa solução além de tornar muito difícil a manutenção da integridade da base de dados, acessada por vários clientes, ainda pode degradar o desempenho da rede.

Em contrapartida aos problemas da primeira forma de acesso a banco de dados, poderíamos pensar em uma solução, onde parte das funções do SGBD fosse executada no servidor, que agora assumiria não apenas as funções de armazenamento de um servidor de arquivos, mas também funções de processamento de consultas, sendo por isso denominado *servidor de bancos de dados*. A utilização de servidores de bancos de dados permite a centralização de funções como controle de concorrência e manutenção de consistência dos bancos de dados, viabilizadas pelo processamento orientado a transações implementado nesses servidores.

Utilizando um servidor de Banco de Dados a um grande aumento do desempenho das aplicações da rede, só para termos uma pequena noção, imagine uma Base de Dados com 1000 cadastros, e o usuário (cliente) quer uma consulta de todos os cadastros que possuem cidade igual a Tubarão. Se utilizarmos o servidor de arquivo, teremos que transferir todos os 1000 cadastros pela rede. Já se utilizarmos o servidor de

Banco de Dados, nós só mandamos (do cliente para o servidor) um pedido em SQL, o servidor faz a filtragem e manda apenas para o cliente os cadastros exigidos.



Dois aspectos explicam o aumento do desempenho da aplicação quando é utilizado um servidor de banco de dados. O primeiro deles é a utilização da rede, que no segundo caso só transporta registros que efetivamente irão compor o relatório. Nesse caso, o ganho no desempenho não se restringe à aplicação em questão, mas estende-se a todas as aplicações que estejam utilizando a rede para intercambiar dados. O Segundo aspecto a ser observado, é a possibilidade de concentrar os investimentos no hardware do servidor (processador, memória cache, etc.), implicando em uma velocidade de processamento maior do que no caso onde a seleção dos registros é realizada no cliente, usualmente uma máquina de menor capacidade de processamento que o servidor. O custo extra do software e hardware do servidor pode ser facilmente justificado pelo seu compartilhamento.

3.4. Sistemas Paralelos

Sistemas paralelos imprimem velocidade ao processamento e à I/O por meio do uso em paralelo de diversas CPUs e discos. Equipamentos paralelos estão se tornando bastante comuns.

No processamento paralelo, muitas operações são realizadas simultaneamente, ao contrário do processamento serial. Um equipamento paralelo de *granulação-grossa* consiste em poucos e poderosos processadores; um *paralelismo intensivo* ou de *granulação-fina* usa milhares de pequenos processadores

3.5. Sistemas Distribuídos

Em um Sistema de Banco de Dados Distribuído, o banco de dados é armazenado em diversos computadores. Os computadores de um Sistema de Banco de Dados distribuído comunicam-se com outros por intermédio de vários meios de comunicação. Eles compartilham memória principal ou discos. Os computadores em um Sistema Distribuído podem variar, quanto ao tamanho e funções, desde estações de trabalhos até sistemas de grande porte.

Lição 04
Banco de Dados Cliente/Servidor

"A razão é Deus, e Deus é racional"

Malebranche

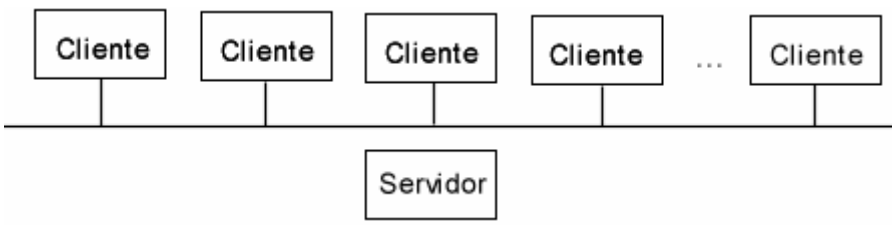
4. Objetivo

O objetivo desta lição é bem simples: entender o Banco de Dados Cliente/Servidor.

4.1. Sistema Cliente/Servidor

Como os computadores pessoais têm se tornado mais rápidos, mais poderosos e baratos, há uma tendência do seu uso nos sistemas centralizados. Terminais conectados a sistemas centralizados estão sendo substituídos por computadores pessoais. Simultaneamente, interfaces com o usuário usadas funcionalmente para manuseio direto com sistemas centralizados estão sendo adequadas ao trato com computadores pessoais.

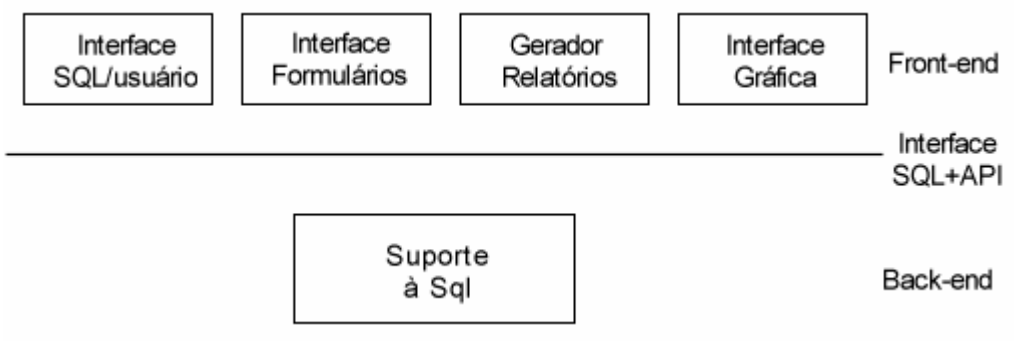
Como resultado, sistemas centralizados atualmente agem como sistemas servidores que atendem a solicitações de sistemas clientes. A estrutura geral:



As funcionalidades de um banco de dados cliente/servidor podem ser superficialmente divididas em duas categorias:

- *back-end* : gerencia as estruturas de acesso, desenvolvimento e otimização de consultas, controle de concorrência e recuperação.
- *front-end* : consiste em ferramentas como formulários, gerador de relatórios e recursos de interface gráfica.

A interface entre o front-end e o back-end é feita por meio da SQL ou de um programa aplicativo.



Sistemas servidores podem ser caracterizados, de modo geral, como servidores de transações e servidores de dados.

- *Sistemas servidores de transações*, também chamados *sistemas servidores de consulta* (query-server), proporcionam uma interface por meio da qual os clientes podem enviar pedidos para determinada ação e, em reposta, eles executam a ação e mandam de volta os resultados ao cliente. Usuários podem especificar pedidos por SQL ou por meio de um programa de aplicação usando um mecanismo de chamada de procedimento remota (remote-procedure-call).
- *Sistemas servidores de dados*, permitem que os servidores interajam com clientes que fazem solicitações de leitura e atualização de dados em unidades como arquivos ou páginas. Por exemplo, servidores de arquivos proporcionam uma interface sistema-arquivo na qual os clientes podem criar, atualizar, ler e remover arquivos. Servidores de dados para sistemas de banco de dados oferecem muito mais recursos: dão suporte a unidade de disco – como páginas, tuplas ou objetos – menores que um arquivo. Proporcionam meios para indexação de dados e transações, tal que os dados nunca se tornem inconsistentes se um equipamento cliente ou processo falhar.

4.1.1. Servidor de Transações

Em sistemas centralizados, o front-end e o back-end são ambos executados dentro de um único sistema. Porém, a arquitetura de servidores de transações segue a divisão funcional entre front-end e back-end.

Normalmente, o front-end fica em computadores pessoais, os quais fazem pedidos ao servidor (back-end), onde está localizado os dados; o back-end retorna a resposta e é de responsabilidade do front-end exibir estas respostas.

Inicialmente, precisávamos que o back-end e o front-end fossem do mesmo fabricantes, pois não tinha um padrão. Hoje, utilizamos o ODBC (open database connectivity), que são programas de aplicação de interface que permite ao cliente criar uma consulta SQL e enviar ao servidor, que retornará uma resposta. Qualquer cliente que utilize uma interface ODBC, pode se conectar a um servidor que de suporte a ODBC.

Com a difusão dos padrões foram criados diversos aplicativos independentes, tais como o Gupta SQL e PowerBuilder que são ferramentas de front-end independentes das back-ends; ainda foram criados aplicativos front-end específicos, tais como, planilhas.

Outra forma de fazer um pedido é enviando uma chamada de procedimento transacional remota (transactional remote procedure call), esta chamada "encapsulada" enviada ao servidor é tratada como uma chamada qualquer, assim se ocorrer alguma falha o servidor reverte toda a chamada.

As vantagens dos bancos de dados cliente/servidor são a maior funcionalidade e o menor custo, maior flexibilidade na disseminação, expansão e alocação dos recursos, melhores interfaces com os usuários e manutenção mais fácil.

4.1.2. Servidores de Dados

Os Servidores de Dados estão em redes de alta velocidade, em que os computadores clientes e servidor tem mesmo potencial. Em tal ambiente, há grande tráfego do servidor para o cliente e do cliente para o servidor. Note que essa arquitetura exige ampla funcionalidade back-end nos clientes. Esta arquitetura esta ganhando espaço nos sistemas de banco de dados orientados a objetos.

O interesse neste tipo de arquitetura é vantajosa quando a transferência é alta em ralação ao consumo de memória principal.

Lição 05
Banco de Dados Distribuídos

"O poeta é o criador da nação em sua volta."

Herder

5. Objetivo

Como na lição anterior, o objetivo desta lição é bem simples: entender o Banco de Dados Distribuídos.

5.1. Distribuídos

Como já falamos, em um Sistema de Banco de Dados Distribuído, o banco de dados é armazenado em diversos computadores. Os computadores de um Sistema de Banco de Dados distribuído comunicam-se com outros por intermédio de vários meios de comunicação. Eles compartilham memória principal ou discos. Os computadores em um Sistema Distribuído podem variar, quanto ao tamanho e funções, desde estações de trabalhos até sistemas de grande porte.

Os sites ou nós, como são chamados os computadores, são fracamente acoplados. Além disso, os sistemas de banco de banco de dados em cada site podem possuir um alto grau de independência mútua.

5.1.1. Armazenamento Distribuído dos Dados

Considere uma relação r armazenada em um banco de dados. Há diversos enfoques para o armazenamento dessas relações em um banco de dados distribuído:

- **Replicação**. O sistema mantém réplicas idênticas da relação. Cada réplica é armazenadas em diferentes sites, resultando na replicação dos dados. A alternativa para replicação é armazenar somente uma cópia da relação r.

- **Fragmentação**. A relação é particionada em vários fragmentos. cada fragmento é armazenado em um site diferente.
- **Replicação e fragmentação**. A relação é particionada em vários segmentos. O sistema mantém diversas réplicas de cada fragmento.

5.1.2. Transparência de Rede

É essencial que o sistema esconda do usuário, detalhes relativos à distribuição dos dados na rede, chamamos isto de transparência de rede.

Consideramos os objetivos da transparência do ponto de vista da:

- Denominação dos dados;
- Replicação dos itens de dados;
- Fragmentação dos itens de dados;
- Locação das réplicas e fragmentos.

5.1.3. Processamento de Consultas Distribuídas

Nos sistemas distribuídos devemos considerar diversos problemas em relação ao processamento de consultas:

- O custo de transmissão de dados na rede;
- O ganho potencial de desempenho diante do fato de que diversos sites podem processar partes da consulta em paralelo.

O custo relativo de transferência de dados na rede e de transferência de dados entre discos varia significativamente, dependendo do tipo de rede e da velocidade dos discos.

5.1.4. Modelo de Transações Distribuídas

Há dois tipos de transações que devemos considerar:

- *locais*: acessam somente a base de dados local;
- *globais*: acessam a base de dados localizados em outros computadores, precisando se preocupar com atualizações nesta base.

5.1.5. Coordenador

Alguns algoritmos utilizados nos Sistemas Distribuídos necessita de um coordenador. Se o coordenador falha devido a uma falha no site no qual está sendo executado, o sistema poderá continuar a execução somente reiniciando um novo coordenador em um outro site. Isso poderá ser feito por meio da manutenção de um backup do coordenador, que estará pronto a assumir a responsabilidade do coordenador se este falhar. Ou o sistema poderá escolher um novo coordenador depois que o original falhou. O algoritmo que determina onde a cópia do coordenador será reiniciada é chamada de algoritmo de *eleição*.

Lição 06
Modelagem de Dados

"O único lugar onde o sucesso vem antes do trabalho é no dicionário."

Albert Einstein

6. Objetivo

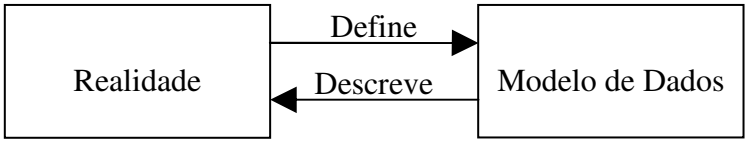
O objetivo desta lição é mostrar o que é e como fazer a modelagem de dados.

O conceito de Modelagem de Dados é simples, Modelagem de Dados é uma técnica de representação da realidade que mostra o conjunto de dados de um determinado ambiente, através de suas entidades e dos relacionamentos existentes entre as mesmas.

Nos Apêndices A, B, C, D e E o aluno encontrará estudos de casos, para treinar.

6.1. O que é Modelo de Dados?

Também conhecido como Diagrama E-R (Entidade -Relacionamento). É uma forma de representação gráfica do conhecimento que se tem sobre o ambiente (realidade) qualquer. Mostra uma visão estática das informações (entidades) de interesse e dos vínculos (relacionamentos) existentes entre elas.



O modelo de dados é uma nova forma de comunicação entre o técnico de processamento de dados e o usuário. Essa nova forma de comunicação assegurará que :

- O modelo de dados conterá todos os dados necessários para suportar os processos de responsabilidade do usuário.

- O modelo de dados conterá os dados para suportar processos que serão modificados ou introduzidos em um futuro próximo.

6.2. Componentes do Modelo de Dados

6.2.1. Entidade

É algo, real ou abstrato, percebido no ambiente e sobre o qual nos interessa armazenar dados.

Exemplos:

Um objeto real (concreto)	- Um equipamento, Material
Uma pessoa	- Fornecedor Empregado
Um conceito abstrato	- Órgão, Cargo, Curso
Um evento	- Recebimento de Material
Um relacionamento	- Casamento

Representação gráfica: Uma entidade é representada num modelo de dados através de um retângulo.

6.2.2. Atributo

É um dos itens de dados que armazenamos sobre uma entidade. Caracteriza ou qualifica uma determinada propriedade de uma entidade.

Exemplo:

São atributos da entidade EMPREGADO:

- MATRICULA
- NOME
- ENDERECO
- CPF
- DATA NASCIMENTO

6.2.3. Chave de Identificação

A chave de identificação de uma entidade é definida por um atributo, ou conjunto de atributos, cujos valores individualizam uma única ocorrência dessa entidade.

Exemplo: A chave de identificação da entidade EMPREGADO é o atributo MATRICULA.

6.2.4. Lista de Entidades

É uma relação de entidades com seus respectivos atributos, utilizada para documentar os trabalhos de análise de dados.

Formada pelo nome da entidade seguida da relação de atributos que compõem entre parênteses, e seguindo a convenção abaixo:

- Cada atributo é separado do outro pelo sinal de adição (+) ;
- O(s) atributo(s) que identificam a entidade devem estar no início da relação e sublinhados;
- O(s) atributo(s) que ocorrem mais de uma vez (repetitivos) são identificados por uma inclusão entre parênteses.

Exemplo:

FATURA (NUMERO_FATURA + CODIGO_FORNECEDOR +(NUMERO_ITEM_FATURA + CODIGO_MATERIAL + QUANTIDADE_MATERIAL + PRECO_UNITARIO + PRECO_ITEM_FATURA) + PRECO_TOTAL_FATURA).

Obs.: Podem haver múltiplos níveis de repetição.

6.2.5. Domínio

São os possíveis valores que um atributo pode assumir.

Exemplo:

SEXO = [M | F]

Sexo pode assumir dois valores M (Masculino) ou F (Feminino);

Nome = literal, cadeia de caracteres

Numero_Fatura = número inteiro

6.2.6. Relacionamento

É uma ligação (vínculo) existente entre duas entidades, que define como uma descreve a outra.

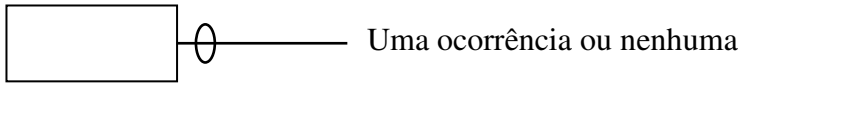
Os relacionamentos são representados por uma linha ligando as entidades inter-relacionadas.

Duas entidades estarão relacionadas entre si, se possuírem, atributos comuns.

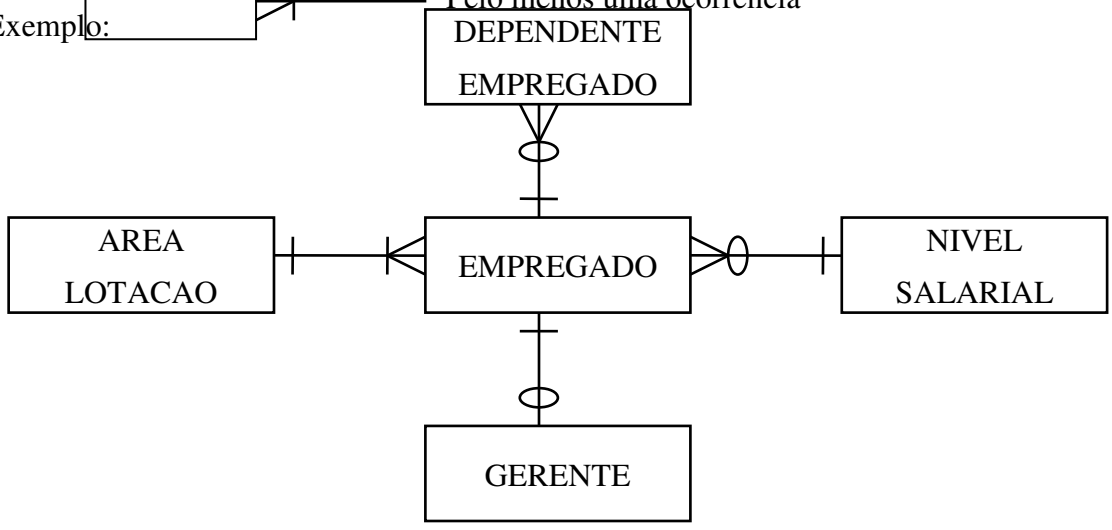
6.2.7. Grau do Relacionamento

São as restrições identificadas na quantidade de ocorrências de uma entidade, que pode estar relacionada a uma ocorrência de outra entidade.

Representação Gráfica: símbolos especiais colocados nas extremidades da linha que representa um relacionamento.



Exemplo:



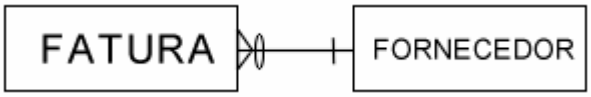
- Uma ÁREA LOTAÇÃO tem obrigatoriamente pelo menos 1 empregado;
- Um EMPREGADO está vinculado obrigatoriamente a uma área de LOTAÇÃO;
- Um EMPREGADO pode ter vários, um ou nenhum DEPENDENTE;
- Um DEPENDENTE (se existir) está obrigatoriamente vinculado a um EMPREGADO.
- Um EMPREGADO pode ser GERENTE.
- Um GERENTE é um EMPREGADO
- Um EMPREGADO tem obrigatoriamente um NÍVEL SALARIAL;
- Em um mesmo NÍVEL SALARIAL podemos ter vários, um ou nenhum EMPREGADO.

6.3. Tipos de Entidade

6.3.1. Entidade Primária

É aquela cuja chave de identificação é feita exclusivamente através de seus atributos. Exemplo:

Exemplo:

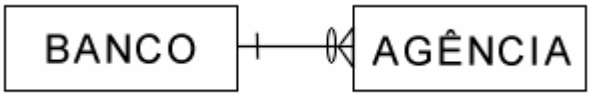


FATURA(Número_Fatura + Cod_Fornecedor+...)

FORNECEDOR(Cod_Fornecedor + Nome_Fornecedor + ...)

6.3.2. Entidade Dependente

É aquela cuja existência depende de outra, ou seja, parte da chave de identificação da entidade está condicionada a da entidade da qual ela depende. Exemplo:



BANCO (Cod_Banco + Nome_Banco)

AGÊNCIA(Cod_Banco + Cod_Agência + Nome_Agência)

6.3.3. Entidade Associativa

É aquela cuja chave de identificação é obtida através da concatenação das chaves de identificação das entidades que ela associa. Exemplo:

DEPOSITO (Cod_Deposito + Localização + ...)

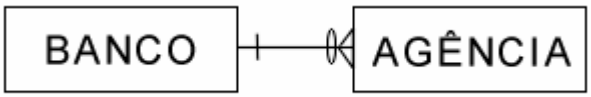
MATERIAL (Cod_Material + Nome_Material + Unidade + ...)

MATERIAL_DEPOSITO (Cod_Deposito + Cod_Material +Quantidade+...)

6.4. Tipos de Relacionamento

6.4.1. Relacionamento de Dependência (D)

É aquele feito entre uma entidade e outra que dela seja dependente. Exemplo:



BANCO (Cod_Banco + Nome_Banco)

AGÊNCIA(Cod_Banco + Cod_Agência + Nome_Agência)

6.4.2. Relacionamento Associativo (A)

É aquele que ocorre entre uma entidade associativa e a cada uma das entidades que participam da associação. Exemplo:

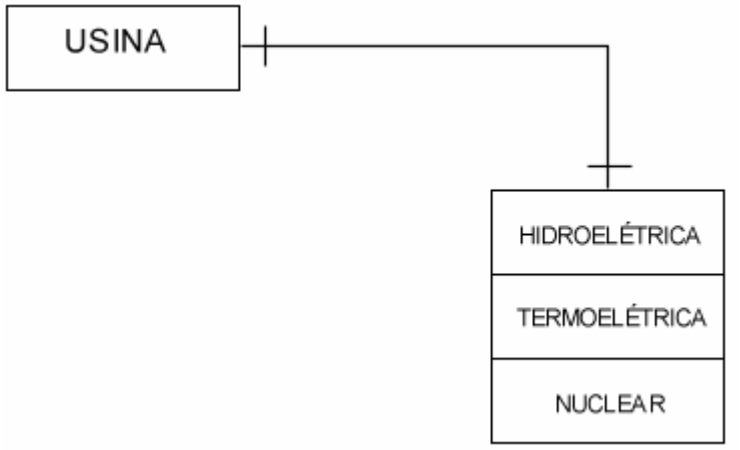
DEPOSITO (Cod_Deposito + Localização + ...)

MATERIAL (Cod_Material + Nome_Material + Unidade + ...)

MATERIAL_DEPOSITO (Cod_Deposito + Cod_Material +Quantidade+...)

6.4.3. Categoria (C)

Uma categoria é uma ligação entre uma entidade e suas espécies (tipos), sendo estas mutuamente excludentes. Exemplo:

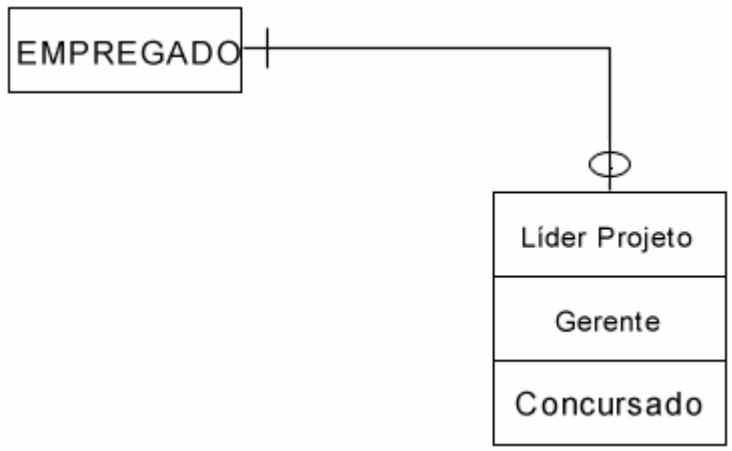


"Uma USINA é uma USINA ELÉTRICA, ou uma USINA TERMOELÉTRICA ou uma USINA NUCLEAR".

Nota: Apesar de constituírem um único tipo de coisa e apresentarem atributos comuns, podem existir atributos específicos para cada espécie.

6.4.4. Partição (P)

É um caso particular de categoria, na qual as espécies (tipos) de uma entidade, não são mutuamente excludentes. Exemplo:



"Um EMPREGADO pode ser um Líder Projeto, e/ou um Gerente, e/ou um Concursado, ou nenhum deles".

6.4.5. Relacionamento Normal

É aquele que não pode ser enquadrado em um dos tipos abaixo:

- Associativo
- Dependência
- Categoria
- Partição

6.4.6. Auto-Relacionamento

É aquele que ocorre entre uma mesma entidade.

6.4.7. Múltiplos Relacionamentos

Casos em que ocorre mais de um relacionamento entre duas mesmas entidades.

6.4.8. Relacionamento Mutuamente Exclusivo

Ocorre quando temos um relacionamento, por exemplo, entre as entidades "A" e "C" e também entre as entidades "B" e "C", porém nunca ao mesmo tempo.

6.5. Tipos de Chave

6.5.1. Chaves Candidatas

São as possíveis chaves de identificação de uma única ocorrência de uma entidade. Exemplo:

EMPREGADO (MATRICULA + NOME + CPF + ENDERECO +...)

São chaves candidatas os atributos:

- MATRICULA
- CPF

6.5.2. Chave Primária

É uma das chaves candidatas, selecionada por melhor conveniência (facilidade de utilização, menor possibilidade de erros, etc. ...).Exemplo:

EMPREGADO (MATRICULA + NOME + CPF + ENDERECO +...)

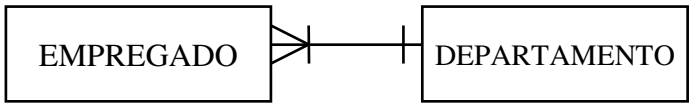
Chaves candidatas:

- MATRICULA
- CPF

Chave Primária escolhida -> MATRICULA

6.5.3. Chaves Estrangeira

Conjunto de um ou mais atributos de uma entidade que são chave primária em outra entidade. Exemplo:



EMPREGADO (MATRICULA + NOME + CPF + COD_DEPTO)

DEPARTAMENTO (COD_DEPTO + NOME_DEPTO)

Na entidade EMPREGADO o atributo "COD_DEPTO" é chave estrangeira.

6.6. Consolidação de Modelos de Dados

6.7.1. O que é Consolidação?

Termo utilizado para representar os trabalhos de integração de um modelo de dados a outro ou, integração de modelos parciais a um modelo global de dados (empresa, assunto ou sistema).

6.6.2. Trabalhos Executados na Consolidação

Os Trabalhos da consolidação basicamente são os seguintes:

- Adição de entidades ainda inexistentes no modelo global de dados, relacionando-as às demais;

- Adição de novos atributos a entidades já existentes, desde que possuam chaves primárias idênticas;
- Identificação das entidades já implementadas;
- Eliminação de relacionamentos redundantes.

Lição 07
Normalização dos Dados

"Deus é, sobretudo, um artista(...)"

Pablo Picasso

7. Objetivo

Esta lição tem como objetivo mostrar o que é e como fazer a Normalização dos Dados, bem como, mostrar a importância deste processo. Nos Apêndices A, B, C, D e E o aluno encontrará estudos de casos, para treinar as técnicas de Normalização.

7.1. O que é Normalização

Normalização é o processo formal que consiste em substituir um conjunto de entidades por outro conjunto capaz de comportar melhor as mudanças futuras.

Entidades normalizadas não possuem redundâncias (duplicação de dados) acidental. Cada atributo está relacionado com sua própria entidade e não se mistura com atributos relativos à entidades diferentes.

A normalização corresponde na realidade à formalização de regras baseadas no fato que as entidades possuem anomalias de atualização.

7.2. Anomalias de Atualização

Dada a entidade:

PEDIDO (numero_pedido + data_pedido + numero_cliente + nome_cliente + endereco_cliente + (numero_produto + nome_produto + qtde_pedida + preco_produto + total_produto) + total_pedido)

Quais as anomalias de atualização que acontecerão se:

- Um produto for descontinuado por seu fornecedor?
- O nome do produto for mudado?
- O cliente mudar de endereço?

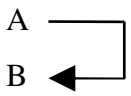
- Os produtos ou as quantidades pedidas pelo cliente forem mudadas e o cliente esqueceu o número do pedido?

7.3. Dependência Funcional

Dados os atributos "A" e "B" de uma entidade, diz-se que "B" é funcionalmente dependente de "A" se e somente se, a cada valor de "A" está associado um único valor de "B".

Em outras palavras, se conhecermos o valor de "A" então podemos encontrar o valor de "B" associado a ele.

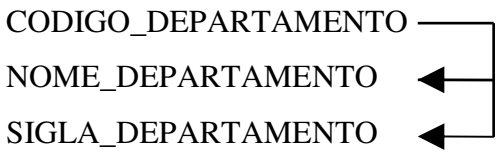
DIAGRAMA DE DEPENDÊNCIA FUNCIONAL



Nota - a seta parte de quem identifica.

Exemplo:

DEPARTAMENTO

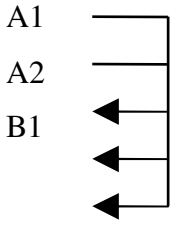


Nota - O exame das relações existentes entre os atributos de uma entidade deve ser feito a partir do conhecimento (conceitual) que se tem sobre o mundo real (ambiente modelado).

7.4. Dependência Funcional Composta ou Completa

Dado um atributo ou um conjunto de atributos "B" de uma entidade, sendo a chave primária composta por um conjunto de atributos "A", diz-se que "B" é completamente dependente funcional da chave primária, se e somente se, a cada valor da chave (e não a parte dele), está associado um valor para cada atributo do conjunto "B".

DIAGRAMA DE DEPENDÊNCIA FUNCIONAL

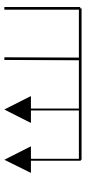


B2
B3

Exemplo:

PRODUTO_FATURA

NUMERO_PEDIDO
CODIGO_PRODUTO
QTDE_PEDIDA
PRECO_TOTAL_PRODUTO

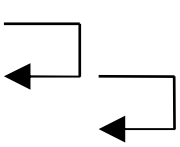


7.5. Dependência Transitiva

Dados os atributos “A”, “B” e “C” de uma entidade, sendo “A” a chave primária, diz-se que “B” e “C” são dependentes transitivos se e somente se, forem funcionalmente dependente de “A” além de existir uma dependência funcional entre eles.

DIAGRAMA DE DEPENDÊNCIA FUNCIONAL

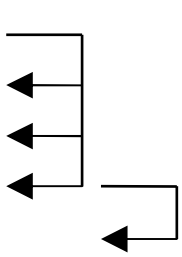
A
B
C



Exemplo:

DEPARTAMENTO

CODIGO_DEPARTAMENTO
NOME_DEPARTAMENTO
SIGLA_DEPARTAMENTO
MATRICULA_GERENTE
NOME_GERENTE



7.6. Primeira Forma Normal (1FN)

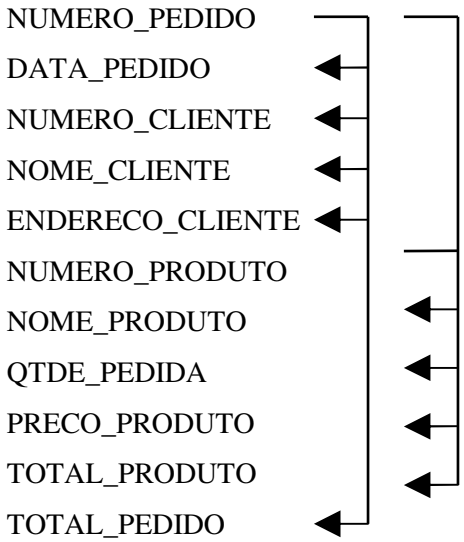
Uma entidade está na 1FN se ela não contém grupos de atributos repetitivos (multivalorados).

Exemplo:

Entidade não normalizada:

PEDIDO (NUMERO_PEDIDO + DATA_PEDIDO + NUMERO_CLIENTE + NOME_CLIENTE + ENDERECO_CLIENTE + (NUMERO_PRODUTO + NOME_PRODUTO + QTDE_PEDIDA + PRECO_PRODUTO + TOTAL_PRODUTO) + TOTAL_PEDIDO)

- Remoção dos grupos de atributos repetitivos (1FN):

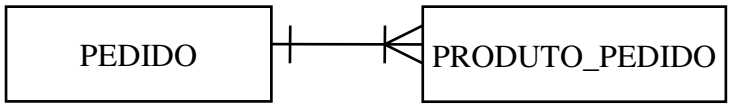


- Entidades da 1FN

PEDIDO (NUMERO_PEDIDO + DATA_PEDIDO + NUMERO_CLIENTE + NOME_CLIENTE + ENDERENCO_CLIENTE + TOTAL_PEDIDO)

PRODUTO_PEDIDO (NUMERO_PEDIDO + NUMERO_PRODUTO + NOME_PRODUTO + QTDE_PEDIDA + PRECO_PRODUTO + TOTAL_PRODUTO)

- Modelo de Dados



7.7. Segunda Forma Normal (2FN)

Uma entidade está na 2FN se ela está na 1FN e seus atributos são funcionalmente dependentes de sua chave (primária) completa.

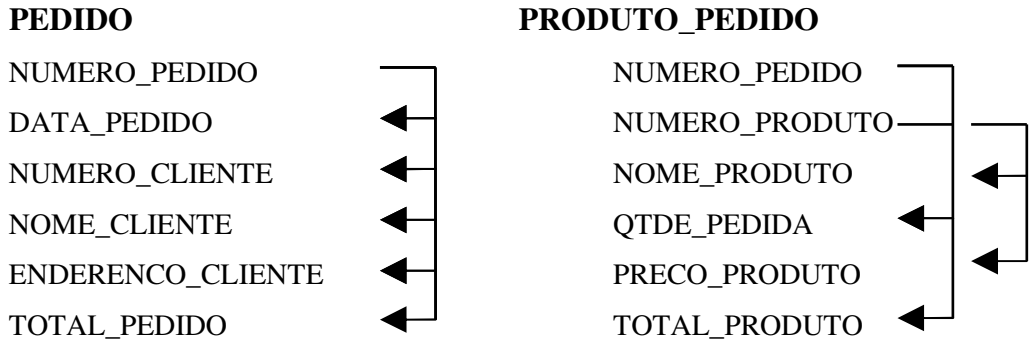
Exemplo:

- Entidades da 1FN

PEDIDO (NUMERO_PEDIDO + DATA_PEDIDO + NUMERO_CLIENTE +
NOME_CLIENTE + ENDERENCO_CLIENTE + TOTAL_PEDIDO)

PRODUTO_PEDIDO(NUMERO_PEDIDO + NUMERO_PRODUTO +
NOME_PRODUTO + QTDE_PEDIDA + PRECO_PRODUTO +
TOTAL_PRODUTO)

- Remoção dos atributos não funcionalmente dependentes de toda uma chave primária (2FN):



- Entidade na 2FN:

PEDIDO (NUMERO_PEDIDO + DATA_PEDIDO + NUMERO_CLIENTE +
NOME_CLIENTE + ENDERECO_CLIENTE + TOTAL_PEDIDO)

PRODUTO_PEDIDO (NUMERO_PEDIDO + NUMERO_PRODUTO +
QTDE_PEDIDA + TOTAL_PRODUTO)

PRODUTO (NUMERO_PRODUTO + NOME_PRODUTO + PRECO_PRODUTO)

- Modelo de Dados:

7.8. Terceira Forma Normal (3FN)

Uma entidade está na 3FN se ela está na 2FN e não possui dependências transitivas. Uma entidade que está na 2FN pode ter um atributo que não é uma chave mas que por si identifica outros atributos. Refere-se a isto como uma dependência transitiva.

Exemplo:

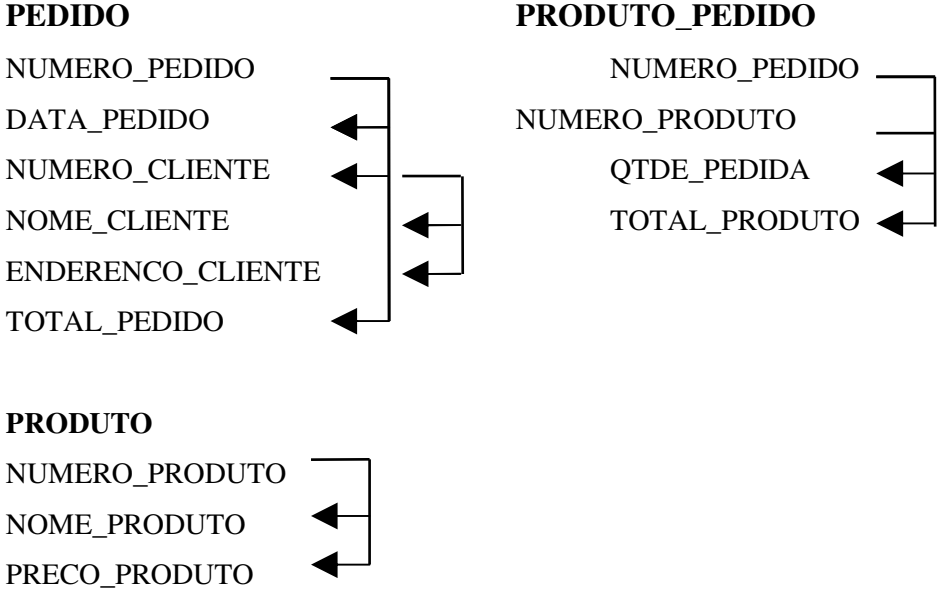
- Entidade na 2FN:

PEDIDO (NUMERO_PEDIDO + DATA_PEDIDO + NUMERO_CLIENTE +
NOME_CLIENTE + ENDERECO_CLIENTE + TOTAL_PEDIDO)

PRODUTO_PEDIDO (NUMERO_PEDIDO + NUMERO_PRODUTO +
QTDE_PEDIDA + TOTAL_PRODUTO)

PRODUTO (NUMERO_PRODUTO + NOME_PRODUTO + PRECO_PRODUTO)

- Remoção das dependências transitivas



- Entidades na 3FN

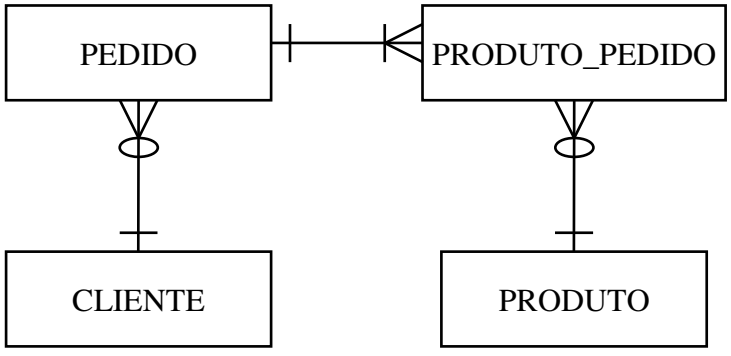
PEDIDO (NUMERO_PEDIDO + DATA_PEDIDO + TOTAL_PEDIDO)

CLIENTE (NUMERO_CLIENTE + NOME_CLIENTE + ENDERECO_CLIENTE)

PRODUTO_PEDIDO (NUMERO_PEDIDO + NUMERO_PRODUTO+
QTDE_PEDIDA + TOTAL_PRODUTO)

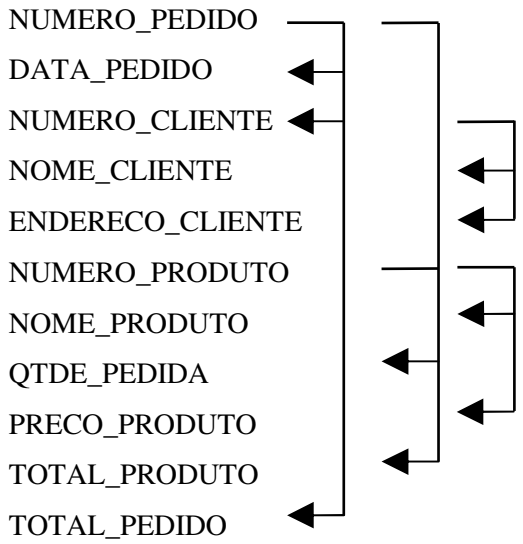
PRODUTO (NUMERO_PRODUTO + NOME_PRODUTO + PRECO_PRODUTO)

- Modelo de Dados.



7.9. Simplificação do Processo de Normalização

A partir do diagrama de dependências funcionais podemos obter diretamente as entidades na terceira forma normal. Para isso, devemos especificar uma entidade para cada conjunto de setas que o diagrama mostrar. A chave primária será formada pelos atributos dos quais partem as setas. Exemplo:



7.10. Regras Práticas

- Se duas entidades possuírem a mesma chave de identificação:
  - Elas são a mesma entidade;
  - Seus atributos se complementam;
  - As suas ocorrências se complementam;
- Quando um atributo ou um conjunto de atributos identificadores de uma determinada entidade, for(em) também atributo(s) de uma outra entidade, deve haver um relacionamento do tipo 1:N entre elas.
- Atributos comuns a mais de uma entidade, devem ser, obrigatoriamente, chaves de identificação em uma das entidades; caso contrário será uma simples redundância.
- Nenhum atributo componente de uma chave primária deve poder assumir um valor nulo. Decorre do fato de que todos os objetos que se quer representar devam ser distinguíveis entre si.
- Um atributo que seja chave estrangeira só pode assumir:

- Valor nulo;
- Valor para o qual exista uma ocorrência da entidade da qual ela é chave primária.

Lição 08
Novas Aplicações

"Os chineses inventaram - e tudo que é bobo repete - que uma imagem vale mais que mil palavras. Diz isso sem palavra!"

Millôr

8. Objetivo

Esta lição se preocupa em mostrar, ao aluno, o que há de novo na área de Banco de Dados. Quais são os novos termos utilizados, também serão mostrados nesta seção.

Os Bancos de Dados Relacionais são aplicados, em grande quantidade, na área comercial, mas quando precisamos de algo fora desta área?

Estudaremos aqui aplicações novas que vêm se tornando cada vez mais importantes.

8.1. Sistemas de Suporte à Decisão

Como a disponibilidade de dados on-line tem crescido, os administradores têm explorado essa fonte de informações para melhorar a tomada de decisão, do tipo: quais itens devem permanecer em estoque e como atender melhor a seus usuários para incrementar as vendas. Podemos extrair a maioria dessas informações de suporte à decisão usando pesquisas simples em SQL. Recentemente, entretanto, esses administradores vêm sentindo a necessidade de um suporte à decisão mais amplo, baseado na análise e extração de dados (data mining), ou na descoberta de conhecimento, utilizando dados de diversas fontes.

8.2. Banco de Dados Espaciais

Banco de Dados Espaciais, também conhecido como Sistema de Informação Geográfica (SIG), são aqueles que nos permitem a captura, modelagem, manipulação, recuperação, análise e apresentação de dados referenciados geograficamente (ou dados georreferenciados).

De forma geral, um software de SIG é um sistema composto de quatro grandes componentes: componente de captura de dados, componente de armazenamento, componente de análise e componente de apresentação dos dados.

8.3. Banco de Dados Multimídia

São Banco de Dados que manuseiam vídeos, imagens e sons. A principal particularidade desse tipo de banco de dados é que os recursos de vídeo e áudio exigem, para exibição dos dados, taxas de resgate de dados estáveis e predefinidas, por isto, são chamados de dados de mídia-contínuos.

8.4. Bancos de Dados Móveis

Muito conhecido pelo termo em inglês “mobile databases”, os hardwares utilizados variam desde notebooks até celulares, sempre usando wireless.

Os computadores devem operar mesmo desconectados da rede, ao contrário dos sistemas de banco de dados distribuídos. Além disso, possuem limites de capacidade de memória e exigem técnicas especiais para seu gerenciamento.

Apêndice A
Estudo de Caso 01

Descrição: Trata-se de um sistema para uma rede hospitalar que atualmente é servida por um sistema manual e desintegrado (pouco provável nos dias de hoje). Feito um levantamento inicial criou-se um modelo de dados preliminar e intuitivo, anexo 01, que serviu de base ao primeiro levantamento. Os dados provenientes deste primeiro levantamento foram organizados nos anexos 02 à 04, de acordo com o modelo de dados preliminar.

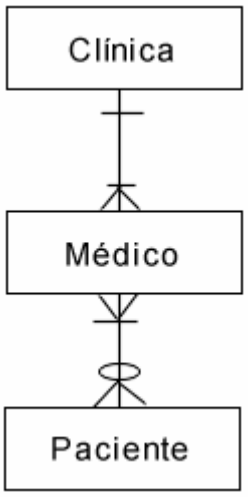
Roteiro: Obter o modelo de dados normalizado até a terceira forma normal, fazendo:

a) diagrama de dependência funcionais;
b) lista de entidades;
c) consolidar lista de entidades;
d) confeccionar o modelo lógico de dados E-R;

Orientações:

- Desenhar um diagrama de dependências funcionais para cada anexo. Isto significa que o diagrama não deverá ultrapassar os limites de um anexo.
- Lembrar que um diagrama de dependências funcionais identifica entidades. Destacar os conjuntos de setas que correspondem a cada uma.
- Evitar sempre considerações que possam corresponder a etapa de desenho físico do banco de dados, tais como: eficiência de acesso e armazenamento, como farei para acessar os dados, etc.

Anexo 01 – Modelo de Dados Preliminar – Rede Hospitalar



Anexo 02 – Dados da "Entidade Paciente"

Nome_Paciente

Endereco_Paciente

Identidade_Paciente

Sexo_Paciente

Data_Nasc_Paciente

Consultas_Marcadas

- Data_Consulta
- Hora_Consulta
- Endereço_Atendimento (local de atendimento ou consultório médico)
- CRM_Médico

Consultas_Realizadas

- Data_Consulta
- Hora_Consulta
- Endereço_Atendimento (local de atendimento ou consultório médico)
- CRM_Médico

Obs.: não há junta médica.

Anexo 03 – Dados da "Entidade Médico"

Nome_Médico

CRM_Médico

Endereço_Médico

Cod_Espec (Código de especialização)

Nome_Espec

Cod_Clinica

Agenda_do_Médico

- Data_Consulta
- Hora_Consulta
- Endereco_Atendimento

Consultas_Realizadas

- Data_Consulta
- Hora_Consulta
- Endereco_Atendimento
- Identidade_Paciente

Anexo 04 – Dados da "Entidade Clínica"

Cod_Clínica

Nome_Clínica

Endereço_Clinica

CGC_Clínica

Locais_de_Atendimento (Grupo que registra os consultórios da clínica)

Endereço_Atend

Cod_Espec

Obs.: Uma clínica, em determinado endereço de atendimento, atende somente uma especialidade; Em locais diferentes, uma clínica pode Ter mais de uma especialidade (ou a mesma); Um único local pode, por meio de um convênio, ser compartilhado por duas ou mais clínicas.

Apêndice B
Estudo de Caso 02

Entrevista com a diretoria de produção:

A diretoria de produção é a responsável pela atividade fim da empresa.

Suas preocupações se resumem em controlar a qualidade dos carros que saem da linha de produção e controlar a produtividade dos operários.

O controle de qualidade se faz por amostragem ao longo da linha de produção. Os carros são inspecionados e o resultado da inspeção é registrado num relatório (Anexo 01). Em média, a cada 1000 carros inspecionados, só 10 apresentam defeitos.

O controle da produtividade dos operários começa pela vigília sobre o cartão de ponto (Anexo 02). Atrasos diários superiores a 15 minutos ou mensais superiores a 6 horas são anotados em sua ficha cadastral, para efeito de análise da validade de futuras promoções. Uma contagem de produção (número de peças produzidas em determinado período) é feita por grupo de operários (anexo 06). Cada grupo produz uma única peça e todos operários, a ele pertencentes, trabalham no mesmo turno. Cada carro saído da linha de produção é registrado conforme Anexo 03.

Entrevista com a diretoria comercial:

Com base nos registros de vendas enviados pelas concessionárias (Anexo 04), a diretoria escalona o encaminhamento dos carros saídos da linha de produção.

Através do Registro de vendas e do registro de manutenção (Anexo 05), a fábrica acompanha a vida de cada carro durante o período de garantia (1 ano).

Cada carro é revisto até três vezes durante o período da Garantia (Manutenção).

Anexo 01 – Relatório de Inspeção

Cod_Turno_Trabalho

Data_Inspeção_Fab

Num_Chassi

Itens_Inspecionados

Cod_Item_Inspecionado

Cod_Status_Inspeção

Anexo 02 – Irregularidade de Freqüência

Mês_Irregularidade

Matrícula_Func

Cod_Grupo_Trab

Nome_Funcionario

Cod_Turno_trabalho

Irregularidades_Freqüência

- Data_Irregularidade
- Tipo_Irregularidade
- Minutos_de_Atraso

Anexo 03 – Registro de Produção

Num_Chassi

Data_Fabricação

Modelo_Carro

Ano_Carro

Cor_Carro

Anexo 04 – Registro de Vendas das concessionárias

Num_Chassi

Modelo_Carro

Ano_Carro

Cor_Carro

Cod_Concessionaria

Nome_Prop_Carro

Forma_Venda

Data_Venda

Anexo 05 – Contagem da Produção

Data_Contagem_Prod

Contagem_Produção

- Cod_Grupo_Trab
- Cod_Peça
- Quant_Peça

Anexo 06 – Registro de Manutenção

Num_Chassi

Data_Manutenção

Cod_Concessionaria

Itens_Inspecionados_Manutenção

- Cod_Item_Inspecionado
- Cod_Status_Inspeção

Peças_Substituição_Manutenção

- Cod_Peça
- Cod_Defeito_Peça
- Cod_Motivo_Defeito_Peça

Apêndice C
Estudo de Caso 03

Projetar uma base de dados - modelo relacional - nível conceitual - para atender a área de compras de uma empresa. Em levantamentos efetuados com o pessoal responsável pelo setor de compras, foram identificados os seguintes informações: E identificar as chaves de cada entidade.

ORDEM DE COMPRA

Código-Ordem-Compra

Data-Emissão

Código-Fornecedor

Nome-Fornecedor

Endereço-Fornecedor

%Materiais da ordem de Compra

codigo-item (n)

descrição-item(n)

valor-unitario-item (n)

quantidade-comprada-item (n)

valor-total-item (n)

valor-total-compra

Procure obter um modelo de dados sem redundâncias, desenhando o diagrama ER.

Apêndice D
Estudo de Caso 04

Projetar uma base de dados - modelo relacional - nível conceitual - para atender a área de recursos humanos de uma empresa. Em levantamentos efetuados foram identificados os seguintes dados:

DADOS FUNCIONÁRIOS

Matricula - Funcionário

Nome - Funcionário

Endereço - Funcionário

Data - Funcionário

Código - Cargo

Valor - Salário

Numero - total - dependentes

Código - departamento

%Habilidades (grupo multivalorado)

  Código - Habilidade (n)

  Descrição - Habilidade (n)

  Data - Formação - Habilidade (n)

% dependentes (grupo multivalorado)

  Código - Dependente

  Nome - Dependente

  Data - nascimento - Dependente (n)

DADOS DEPARTAMENTO

Código - Departamento

Nome - Departamento

Localização - Departamento

Procure definir um modelo de dados sem redundâncias.

Apêndice E
Estudo de Caso 05

Projetar uma base de dados - modelo relacional - nível conceitual - para atender necessidades de um candidato ao governo do estado. Foram identificados os seguintes informações:

CIDADES

Código-Cidade(CEP)

Nome-Cidade

População-Cidade

Prefeito-Atual-Cidade

Partido-Prefeito-Atual

%Zonas (n)

  Numero-Zona (n)

  Local-Zona (n)

  Número-Eleitores-Zona (n)

  Cabo-Eleitoral-Principal(n)

VEREADORES

Código-Vereador

Nome-Vereador

Código-Cidade

Nome-Cidade

Partido-Vereador

Voto-Ultima-Eleição

Mandato-Vereador(periodo)

DEPUTADOS

Código-Deputado

Nome- Deputado

Código-Cidade

Nome-Cidade

Voto-Ultima-Eleição

Partido-Deputado

Mandato-Deputado(periodo)

Categoria-Deputado(Estadual-Federal)

PRINCIPAIS SOLICITAÇÕES CIDADES

Código-Cidade

Nome-Cidade

%Solicitações

Número-Solicitação (n)

Descrição-Solicitação(n)

Data-Solicitação (n)

Viabilidade-Atendimento(n)

Orgãos-Envolvidos(n)

Proposta:

- Obter E-R normalizado até a 3ª forma normal.
- Demonstrar as normalizações efetuadas via esquema de dependências funcionais
- Listar suposições que julgar necessárias sobre a semântica das dependências envolvidas
- Explicar campos de relacionamentos
- Se julgar necessário inclua novos campos e justifique a inclusão.

Bibliografia

KORTH, Henry F. **Sistema de bancos de dados /** Henry F. Korth, Abraham Silberschatz; tradução Maurício Heihachrio Galvan Abe; revisão técnica Sílvio Carmo Palmieri – 2 ª ed. – São Paulo: Makron Books, 1993.

KORTH, Henry F. **Sistema de bancos de dados /** Henry F. Korth, Abraham Silberschatz, S. Sudarshan; tradução Marília Guimarães Pinheiro, Cláudio César Canhette; revisão técnica Luis Ricardo de Figueiredo – 3 ª ed. – São Paulo: Makron Books, 1999.

VIESCAS, John. **SQL: a linguagem padrão de Banco de Dados relacionais: Quick Reference /** John Viescos; Tradução de Dalton Conde de Alencar – Rio de Janeiro: Campus, 1989.

BYRNE, Jeffry. **Microsoft Access 97-Rápido e Fácil para Iniciantes /** Jeffry Byrne; Tradução de Elisa M Ferreira – Rio de Janeiro: Campus, 1997.

Escola de Informática da SBC Sul. **Livro Texto**./editado por Raul Ceretta Nunes – Santa Maria: Departamento de Eletrônica e Computação da UFSM, 2000. ERI 200, 2000.