FACULDADE DE TECNOLOGIA DE SÃO PAULO

Elisângela Rocha da Costa

BANCOS DE DADOS RELACIONAIS

São Paulo
2011

ELISÂNGELA ROCHA DA COSTA

Bancos de Dados Relacionais

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de São Paulo, como requisito parcial para a obtenção do grau de Tecnólogo em Processamento de Dados.

Orientador: Prof. Paulo Roberto Bernice

São Paulo
2011

AGRADECIMENTOS

Ao Prof. Bernice, meu orientador, pelo apoio na elaboração deste trabalho.

RESUMO

Atualmente, obter informação rápida e confiável é vital para a sociedade, sobretudo para as organizações. Bancos de dados possibilitam o controle e a disponibilização dessas informações e por isso tornaram-se elementos indispensáveis. Desta forma, o presente trabalho se propõe a abordar alguns dos principais tópicos dessa área. Todavia, dada a riqueza do tema e, consequentemente, sua extensão, focalizou-se apenas os assuntos que dizem respeito a bancos de dados relacionais. Para tanto, foi realizada uma pesquisa bibliográfica em algumas obras de referência. O texto a seguir elaborado discute, entre outros, os conceitos de bancos de dados, sistemas de banco de dados, modelagem semântica, modelo de dados, modelo relacional e linguagem de consulta estruturada. Ao final, demonstra-se através de um estudo de caso a importância da abordagem de banco de dados. Ali, também fica ratificado que a compreensão de assuntos concernentes ao campo de banco de dados propicia o desenvolvimento de projetos de banco de dados mais eficientes.

Palavras-chave: bancos de dados, bancos de dados relacionais, armazenamento de dados.

ABSTRACT

Currently, to obtain information quickly and reliably is vital to society, especially for organizations. Databases allow the control and the availability of such information and therefore have become indispensable elements. Thus, this study aims to address some of the main topics in that area. However, given the richness of the subject and, consequently, its length is only focused on the issues that relate to relational databases. To this end, a literature search was performed in some reference works. The following text discusses prepared, among others, the concepts of databases, database systems, semantic modeling, data model, relational model and structured query language. At the end, it is demonstrated through a case study approach to the importance of the database. There is also ratified that understanding of issues relevant to the field of the database enables the development of database projects more efficient.

Keywords: databases, relational databases, data storage.

SUMÁRIO

1. INTRODUÇÃO

Os bancos de dados estão cada vez mais presentes em nosso dia-a-dia, visto que a maioria das atividades que realizamos envolvem, direta ou indiretamente, o uso de uma base de dados. Diante disso, apresentaremos nas seções seguintes uma introdução aos conceitos fundamentais de banco de dados. Considerando que os bancos de dados são em sua grande maioria ainda relacionais, esse trabalho os enfatizará. Assim, o primeiro capítulo conceitua banco de dados e também discute as vantagens advindas dessa abordagem. O segundo capítulo define sistema de banco de dados, assim como discorre brevemente sobre cada elemento que o compõe. O terceiro capítulo descreve abstração e modelo de dados para então discorrer sobre a arquitetura para sistemas de banco de dados ANSI/SPARC. O quarto capítulo oferece uma visão geral das etapas de desenvolvimento de um projeto de banco de dados. No quinto e no sexto capítulo é abordado o Modelo Entidade-Relacionamento, bem como a técnica de diagramação correspondente, que é usado para modelar base de dados. O sétimo capítulo apresenta o Modelo Relacional. O oitavo capítulo contém relato sobre o padrão-SQL, linguagem usada para estruturar e manipular banco de dados relacionais. O último capítulo apresenta um sistema gerenciador de banco de dados relacional - Access - e o seu emprego para organizar dados em uma instituição.

2. BANCOS DE DADOS

A expressão Banco de Dados originou-se do termo inglês Databanks. Este foi trocado pela palavra Databases – Base de Dados – devido possuir significação mais apropriada (SETZER; CORRÊA DA SILVA, 2005, p. 1).

Mas afinal, o que é um banco de dados?

Segundo DATE (2004, p. 10), “Um banco de dados é uma coleção de dados persistentes, usada pelos sistemas de aplicação de uma determinada empresa”.

Em outras palavras, um banco de dados é um local onde são armazenados dados necessários à manutenção das atividades de determinada organização, sendo este repositório a fonte de dados para as aplicações atuais e as que vierem a existir.

Para ELMASRI e NAVATHE (2011, p. 3), na expressão Banco de Dados estão subentendidas as propriedades abaixo:

> Um banco de dados representa algum aspecto do mundo real, às vezes chamado de minimundo ou de universo de discurso (UoD – Universe of Discourse). As mudanças no minimundo são refletidas no Banco de Dados.
> Um banco de dados e uma coleção logicamente coerente de dados com algum significado inerente. Uma variedade aleatória de dados não pode ser corretamente chamada de banco de dados.
> Um banco de dados é projetado, construído é populado com dados para uma finalidade específica. Ele possui um grupo definido de usuários e algumas aplicações previamente concebidas nas quais esses usuários estão interessados.

Assim, um banco de dados é um conjunto organizado de dados relacionados, criado com determinado objetivo e que atende uma comunidade de usuários.

2.1. PORQUE UTILIZÁ-LOS

Ao armazenarmos dados em um computador podemos fazê-lo de duas maneiras: utilizando bancos de dados, ou então, arquivos de dados permanentes. A abordagem de banco de dados é considerada a melhor forma, porque apresenta as seguintes vantagens:

- *Controle centralizado de dados*: os dados estão concentrados em um único local e isto proporciona um maior controle. Na abordagem de processamento de arquivos os dados estão dispersos, pois cada aplicação mantém arquivos de dados próprios.
- *Controle da redundância, redução do espaço de armazenamento e compartilhamento de dados*: no processamento de arquivos convencional existe um desperdício do espaço de armazenamento, visto que uma mesma informação geralmente aparece em muitos arquivos diferentes. No enfoque de banco de dados o dado é armazenado apenas uma vez e pode ser compartilhado (de forma concorrente ou não) por diversos usuários.
- *Eliminação de inconsistências e garantia de integridade*: no método tradicional, baseado em arquivos, dada a repetição de informação armazenada, pode acontecer de um mesmo dado apresentar valores divergentes. Isso ocorre, por exemplo, quando um dado que está presente em dois arquivos é atualizado em apenas um local. Diz-se que os arquivos estão inconsistentes, pois apresentam entradas diferentes para um mesmo dado. E se falta consistência, não há integridade (o arquivo possui informações incorretas). Em banco de dados é possível manter a consistência e a integridade dos dados.
- *Estabelecimento de padrões e facilidade de acesso aos dados*: na abordagem de banco de dados, devido à centralização dos dados, torna-se mais propício instituir padrões de nomenclatura e documentação. Devido a essa padronização a recuperação de informação é mais eficiente. Na forma convencional de armazenamento, os dados estão espalhados em arquivos de diversos formatos e as aplicações que acessam esses dados foram escritas em linguagens de programação diferentes.
- *Independência de dados*: no sistema de arquivos, a definição da estrutura de armazenamento e do método de acesso aos dados está inclusa no código das aplicações. Essas são chamadas de dependentes de dados, visto que é impossível alterar a estrutura dos arquivos de dados sem modificar o respectivo programa de aplicação. Bancos de dados, porém, possibilitam a independência de dados, pois permitem a abstração de dados (que será discutido mais adiante).

3. SISTEMA DE BANCO DE DADOS

De acordo com DATE (2004, p. 6), um sistema de banco de dados é “um sistema computadorizado cuja finalidade geral é armazenar informações e permitir que os usuários busquem e atualizem essas informações quando as solicitar”. Para o autor um sistema de banco de dados é composto por *dados*, *hardware*, *software* e *usuários*.

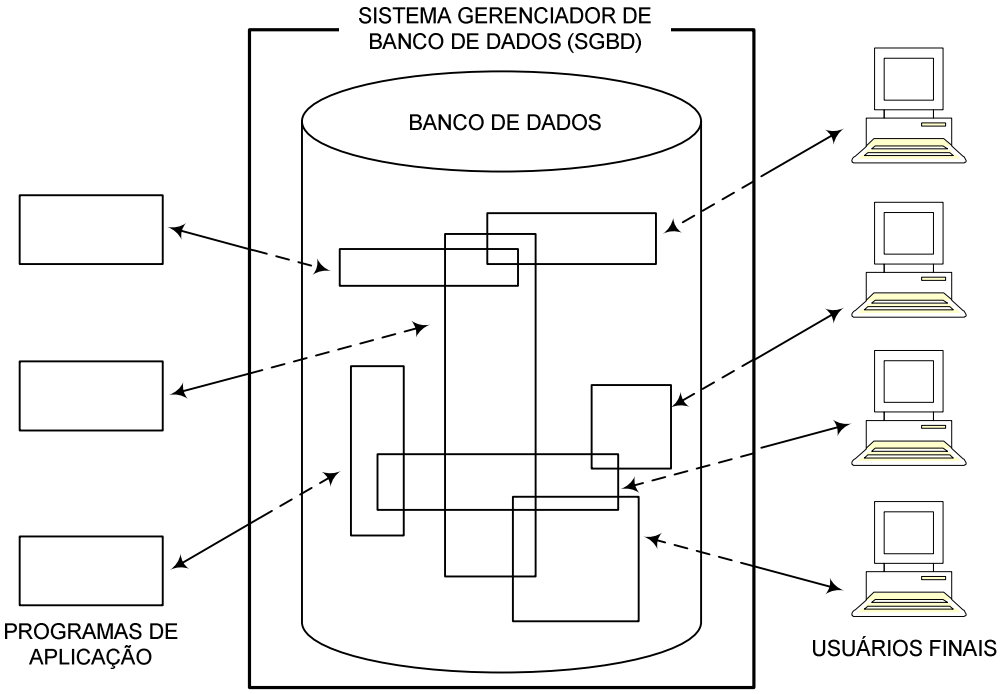


FIGURA 1 REPRESENTAÇÃO DE UM SISTEMA DE BANCO DE DADOS
DATE, 2004, P. 6 (ADAPTADO)

3.1. DADOS

No contexto abordado esse item refere-se ao próprio banco de dados – ao conjunto de dados. Cabe aqui, porém, a conceituação do termo *dado* não realizada anteriormente.

SETZER e CORRÊA DA SILVA (2005, p. 2), definem dado “como uma representação simbólica (isto é, feita por meio de símbolos), quantificada ou quantificável”.

Complementando, ELMASRI e NAVATHE (2011, p. 3), afirmam que dados são “fatos conhecidos que podem ser registrados e possuem significado implícito”.

Desta forma, o número de matrícula de um funcionário é um dado, pois para a empresa ele é um fato conhecido, contém uma semântica, é representado por símbolos (algarismos de 0 a 9 – sistema numérico de base 10, logo é quantificado) sendo, portanto, passível de registro.

3.2. HARDWARE

Compreendem os elementos físicos que compõe o sistema de banco de dados, como as mídias de armazenamento, os canais de entrada/saída, entre outros.

3.3. SOFTWARE

Entre o banco de dados armazenado e os usuários há um conjunto de programas denominado Sistema Gerenciador de Banco de dados (SGBD) (DATE, 2004, p. 8).

“O principal objetivo de um SGDB é proporcionar um ambiente tanto *conveniente* quanto *eficiente* para a recuperação e armazenamento das informações do banco de dados “ (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 1).

Para tanto, o Sistema Gerenciador de Banco de Dados disponibiliza recursos para definir, construir, manipular, compartilhar, proteger e manter bancos de dados (ELMASRI; NAVATHE, 2011, p. 3).

Por *definição* entende-se a especificação das estruturas de armazenamento (definição dos elementos e respectivos tipos de dados que comporão os registros). *Construção* envolve o armazenamento dos dados (registros e relacionamentos). *Manipulação* refere-se à recuperação e a atualização – inclusão, exclusão e alteração – de dados. Por *compartilhamento*, a permissão de acesso concorrente a

uma base de dados. *Proteção*” diz respeito à segurança contra falhas de hardware, software e contra acesso não autorizado. Por *manutenção*, o suporte para o crescimento do banco de dados (ELMASRI; NAVATHE, 2011, p. 4).

3.4. USUÁRIOS

Alguns usuários estão interessados no conteúdo do banco de dados, pois necessitam dos dados lá armazenados para desenvolverem suas atividades diárias. Outros, porém, tem contato com o banco apenas para manter o sistema funcionando corretamente (ELMASRI; NAVATHE, 2011, p. 9).

Destarte, SILBERSCHATZ, KORTH e SUDARSHAN (1999, p. 15), concluíram que os usuários podem ser “diferenciados por suas expectativas de interação com o sistema”.

Segundo DATE (2004, p. 9), há três categorias de usuário:

- *Programador de aplicação*: desenvolve programas sobre o banco de dados, ou seja, cria aplicações que acessarão o sistema de banco de dados;
- *Usuário final*: público que consulta e atualiza o banco de dados utilizando-se, geralmente, das aplicações desenvolvidas pelos componentes da classe de usuários anterior. Pode ter conhecimentos da área de tecnologia da informação (TI);
- *Administrador de banco de dados (DBA)*: responsáveis por gerir o SGDB.

4. ARQUITETURA DE SISTEMAS DE BANCO DE DADOS

Além das características já comentadas anteriormente, a abordagem de banco de dados também permite o que chamamos de *abstração de dados*. Segundo ELMASRI e NAVATHE (2011, p. 19), a abstração de dados “refere-se à supressão de detalhes da organização e armazenamento de dados, destacando recursos essenciais para um melhor conhecimento desses dados”. Em outras palavras, é possível descrever o banco de dados sem ater-se a especificidades de hardware e software.

Para descrever banco de dados não se prendendo a detalhes da forma como ele será implementado utilizamos *modelos de dados*. Na definição de ELMASRI e NAVATHE (2011, p. 19), um modelo de dados é “uma coleção de conceitos que podem ser usados para descrever a estrutura de um banco de dados”. Ainda conforme os autores, de acordo com o tipo de conceito utilizado nessa descrição, os modelos de dados podem ser classificados em:

- *Modelo de dados de alto nível ou conceitual*: é o mais próximo do usuário final. Entidades, atributos e relacionamentos são alguns dos conceitos utilizados. Um exemplo deste modelo é o Modelo Entidade-Relacionamento que será discutido mais adiante.
- *Modelo de dados representativo ou de implementação*: os Modelos de Dados Relacional, Hierárquico e de Rede são exemplares deste modelo. Aqui os dados são mostrados usando estrutura de registro.
- *Modelo de dados de baixo nível ou físico*: descreve os dados do modelo anterior para armazenamento no computador. Trata, por exemplo, do formato dos registros e dos caminhos de acesso a esses dados.

4.1. ESQUEMAS E INSTÂNCIAS

“O conjunto de informações contidas em determinado banco de dados, em um dado momento, é chamado *instância* do banco de dados” (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 6). Ou seja, uma instância é o estado atual do banco de dados. É como uma “fotografia” da base de dados em determinado momento. Ela tende a ser alterada com muita freqüência, pois a cada atualização do banco de dados – inserção, alteração ou exclusão de dados – obtém-se um novo estado.

A descrição do banco de dados denomina-se *esquema*. Esse é definido na fase de projeto do banco de dados e, geralmente, sofre pouca mudança (ELMASRI; NAVATHE, 2011, p. 21). Um esquema pode usar qualquer uma das categorias de modelos de dados vistas acima.

Fazendo uma analogia com os fundamentos das linguagens de programação, diz-se que o valor de uma variável em um determinado instante corresponde à instância do banco de dados. Já a definição do tipo dessa variável corresponde ao esquema de banco (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 6).

4.2. ARQUITETURA DE TRÊS ESQUEMAS

A arquitetura de três esquemas, também conhecida como arquitetura ANSI/SPARC, é uma arquitetura para sistemas de banco de dados cujo objetivo é viabilizar as características da abordagem de banco de dados, sobretudo a independência de dados (ELMASRI; NAVATHE, 2011, p. 22). Ela é composta é composta pelos níveis: *interno*, *externo* e *conceitual*.

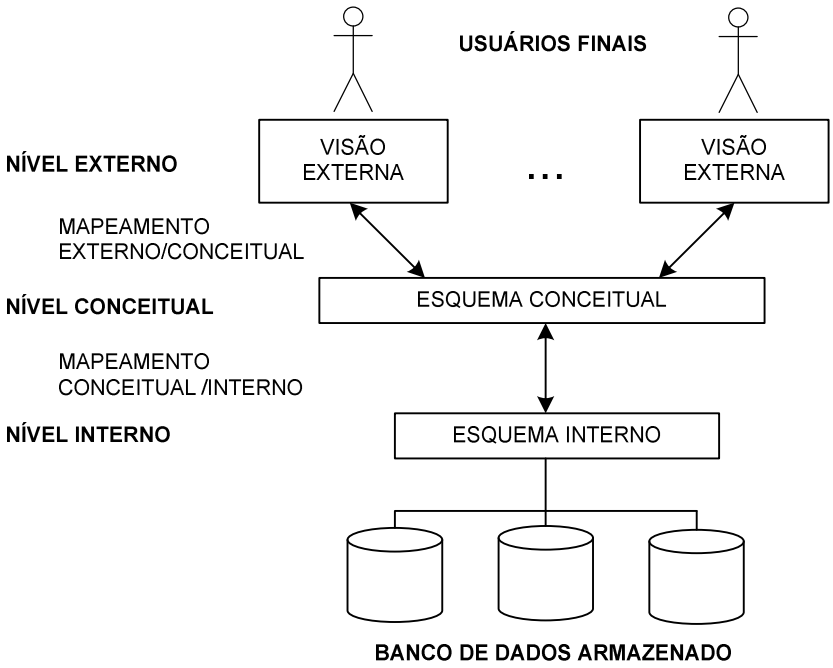


FIGURA 2 ARQUITETURA DE TRÊS ESQUEMAS
ELMARI E NAVATHE, 2011, P. 22 (ADAPTADO)

De forma geral, o usuário está interessado em apenas uma parte do banco de dados. Então, é fornecido a este usuário o acesso a apenas uma porção dos dados armazenados. Essa parcela de dados visualizada por determinado usuário chama-se *visão externa*. As visões externas são definidas através de *esquemas externos*. O *nível externo* ocupa-se desses esquemas.

O *nível conceitual* trata da representação de toda a informação armazenada do banco de dados – *visão conceitual*. Diz-se que é a conjunção dos vários esquemas externos existentes. A visão conceitual é definida através de um *esquema conceitual*.

Já o *nível interno* refere-se à representação da estrutura do armazenamento físico dos dados – *visão interna*. A visão interna é definida por um *esquema interno*.

A figura abaixo exemplifica os três níveis da arquitetura.

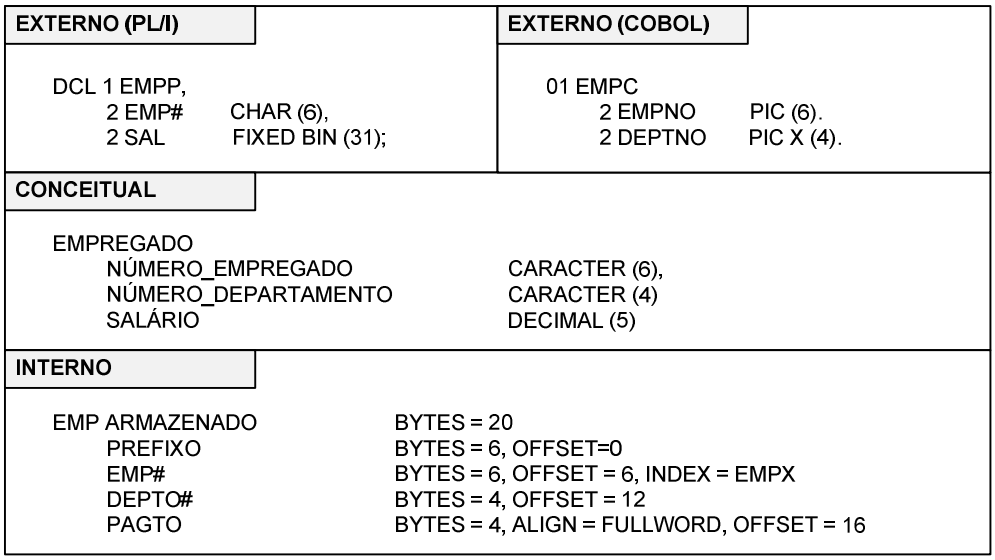


FIGURA 3 EXEMPLO DOS TRÊS NÍVEIS
DATE, 1999, P. 30 (ADAPTADO)

No nível conceitual, o banco de dados armazena informações relativas ao conjunto de entidades EMPREGADO. Cada ente deste conjunto possui três atributos: NÚMERO DO EMPREGADO, NÚMERO DO DEPARTAMENTO E SALÁRIO, sendo já discriminados seus respectivos tipo e tamanho.

No nível externo, existem duas visões - usuário PL/I e usuário COBOL. Em ambas as visões a entidade EMPREGADO é representada por meio de um registro contendo dois campos. Porém, o primeiro usuário enxerga o NÚMERO DO EMPREGADO e o SALÁRIO. Já o segundo vê apenas o NÚMERO DO EMPREGADO e o NÚMERO DO DEPARTAMENTO. Em cada uma das visões o registro é definido conforme convenciona a linguagem de programação utilizada pelo usuário.

No nível interno, a representação é feita por meio de um registro armazenado chamado EMP_ARMAZENADO que contém 20 bytes de comprimento. Esse registro possui, entre outras características, indexação sobre o campo EMP#.

Outra questão importante da arquitetura de três esquemas são os *mapeamentos*. Denomina-se mapeamento a operação de conversão de uma solicitação ou resultado entre os níveis da arquitetura (ELMASRI; NAVATHE, 2011, p. 23). Por exemplo, o usuário que deseja fazer uma consulta especificará sua requisição no respectivo esquema externo. Essa será transformada numa solicitação no esquema conceitual – mapeamento externo/conceitual – que por sua vez terá uma correspondente no esquema interno – mapeamento conceitual/interno. No caminho

inverso, a resposta passará novamente por mapeamentos – interno/conceitual e conceitual/externo – para que seja adequada à visão externa deste usuário.

5. CICLO DE VIDA E PROJETO DE BANCO DE DADOS

Os sistemas de banco de dados, tal qual um software de aplicação, possuem um ciclo de vida. ELMASRI e NAVATHE (2011, p. 204) particionam as atividades desse ciclo em oito fases:

- *Definição do sistema*, corresponde à determinação do escopo do sistema, ou seja, é a decisão sobre principais usuários, quais dados devem ser armazenados e as operações a serem realizadas sobre eles;
- *Projeto do banco de dados*, envolve criação dos *projetos conceitual*, *lógico* e *físico*. No projeto conceitual desenvolve-se um esquema conceitual, utilizando um modelo de dados de alto nível, como por exemplo o Modelo Entidade-Relacionamento, de forma a atender os requisitos apontados durante a fase de definição do sistema. No projeto lógico, também conhecido como mapeamento do modelo de dados, efetua-se uma conversão do esquema criado com o modelo de dados conceitual para o modelo de dados utilizado pelo "tipo" de SGDB que será adotado. Por "tipo" entenda-se o modelo de dados – relacional, hierárquico, rede – adotado pelo SGDB e não o produto específico – Oracle, DB2, entre outros. E finalmente, no projeto físico, este sim dependente do produto SGBD escolhido, são especificadas as estruturas de armazenamento, os índices e caminhos de acessos à base de dados;
- *Implementação do banco de dados*, refere-se à criação do banco de dados de fato, conforme esquemas definidos na etapa anterior;
- *Carga ou conversão de dados*, compreende o preenchimento do banco de dados – povoamento da base. Pode ocorrer pela carga de dados direta ou pela conversão de arquivos subsistentes;
- *Conversão de aplicação*, diz respeito a prováveis adaptações a serem realizadas nos programas que acessavam o sistema anterior para que eles interajam com o novo;
- *Teste e validação*, trata de confrontar o sistema com suas especificações, ou seja, verificar se tudo está funcionando em conformidade com o que foi planejado;
- *Operação*, é relativo à disponibilização do sistema para uso; e

- *Monitoramento e manutenção*, corresponde a observação e a realização de possíveis ajustes do sistema.

▪

6. O MODELO ENTIDADE-RELACIONAMENTO (E-R)

A abordagem de Entidade-Relacionamento é baseada no Modelo Entidade-Relacionamento que foi introduzido por Peter Pin-Shan Chen, em 1976. É um aprimoramento do modelo originalmente proposto, sendo uma das técnicas de modelagem semântica mais conhecidas e, possivelmente, uma das mais utilizadas. DATE (2004, p. 355).

Uma das principais vantagens – talvez seja o motivo maior para sua popularidade – é que além de conceitos o modelo ainda conta com uma técnica de diagramação. Isto permite registrar e comunicar de forma simplificada os principais aspectos do projeto de banco de dados DATE (2004, p. 358).

"O modelo ER descreve os dados como *entidades, relacionamentos* e *atributos*" (ELMASRI; NAVATHE, 2011, p. 132).

6.1. ENTIDADES E CONJUNTO DE ENTIDADES

"Uma entidade é uma 'coisa' ou um 'objeto' no mundo real que pode ser identificada de forma unívoca em relação a todos os outros objetos" (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 21).

Por exemplo, cada servidor de uma instituição pública de ensino é uma entidade. Cada unidade de ensino (*campus*) desse órgão também.

As entidades classificam-se em: *entidades regulares ou fortes* e *entidades fracas*.

Para DATE (2004, p. 355), uma entidade fraca é "uma entidade cuja existência depende de alguma outra entidade, no sentido de que ela não pode existir se essa outra entidade também não existir". Os dependentes de um servidor são exemplos clássicos de entidades fracas, pois existirão se, e somente se, existir a entidade servidor.

Já uma entidade regular ou forte, pode ser definida como uma entidade não fraca. Por exemplo, um servidor é uma entidade forte.

Segundo SETZER e CORRÊA DA SILVA (2005, p. 22), um conjunto de entidades é “uma coleção de entidades que têm características semelhantes, isto é, de entes de uma mesma categoria”.

Assim, o conjunto de entidades Servidores representa a coleção de todos os servidores que trabalham naquela instituição. E o conjunto de entidades Campi refere-se ao conjunto de todas as unidades de ensino daquele órgão.

Um conjunto de entidades é representado no modelo E-R por um retângulo.

FIGURA 4 REPRESENTAÇÃO GRÁFICA DE ENTIDADES

6.2. ATRIBUTOS E DOMÍNIO DE VALORES

“Uma entidade é representada por um conjunto de *atributos*. Atributos são propriedades descritivas de cada membro de um conjunto de entidades” (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 22).

Em outras palavras, atributos são os dados que se deseja guardar sobre cada entidade (SETZER; CORRÊA DA SILVA, 2005, p. 23).

Desta forma, o nome, o prontuário e a data de nomeação são possíveis atributos para cada entidade do conjunto de entidades Servidores. Endereço e sigla comporiam os atributos de cada ente do conjunto de entidades Campi.

Os atributos podem ser classificados como:

- *Simples ou Compostos*

De acordo com ELMASRI e NAVATHE (2011, p. 153), “atributos não divisíveis são chamados **atributos simples** ou **atômicos**”.

Por outro lado, atributos compostos não possuem valor elementar e podem ser decompostos em outros atributos simples e/ou compostos (SETZER; CORRÊA DA SILVA, 2005, p. 24).

Por exemplo, o endereço de cada servidor é um atributo composto, pois pode ser dividido em alguns atributos simples, como: Logradouro, Número, Complemento, Bairro, Cidade e Estado.

- *Monovalorados ou Multivalorados*

Um atributo monovalorado é aquele que assume um único valor para uma dada entidade. Ao passo que, um atributo multivalorado pode ter n valores considerando uma mesma entidade (SETZER; CORRÊA DA SILVA, 2005, p. 27).

Exemplificando, o atributo Sexo do conjunto de entidades Servidores é um atributo monovalorado, pois assume um único valor – masculino ou feminino – para cada entidade. Ao contrário, o atributo Telefone é considerado multivalorado, visto que um servidor pode ter vários telefones para contato e, consequentemente, esse atributo assumirá n valores.

- *Armazenados ou Derivados*

Quanto ao atributo derivado SILBERSCHATZ, KORTH e SUDARSHAN (1999, p. 24), definem que “o valor desse tipo de atributo pode ser derivado de outros atributos ou entidades a ele relacionados”.

O atributo Data_Exercicio que representa a data de entrada em efetivo exercício de cada servidor é um exemplo de atributo armazenado. Já o atributo Tempo_Contribuição que simboliza o tempo total de serviços prestados à instituição é um atributo derivado, porque pode ser obtido pelo valor de Data_Exercício e pela data atual.

- *Nulos*

“Um atributo *nulo* é usado quando uma entidade não possui valor para determinado atributo” (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 24).

O atributo Número_Reservista do conjuntos de entidades Servidores é um atributo nulo, pois não se aplica a todas entidades (servidoras não possuem Carteira de Reservista).

- *Chaves ou Determinantes*

SETZER e CORRÊA DA SILVA (2005, p. 31), definem que “dado um conjunto de entidades, não há duas entidades desse conjunto com o mesmo valor para aquele atributo. Em outras palavras, dado um valor para esse atributo, esse valor determina a qual entidade ele está associado”.

Por exemplo, Prontuário é um atributo chave, pelo motivo de identificar de forma unívoca cada ente no conjunto de entidades Servidores. Não há dois servidores com um mesmo número de prontuário.

Ainda sobre atributos, outro ponto importante é o que se denomina *domínio de valores*. Um domínio de valores diz respeito ao conjunto de valores que determinado atributo pode assumir para cada entidade. Ou seja, conforme esclarece DATE (2004, p. 356), um atributo "tira seus valores de um **conjunto de valores** correspondente (isto é, domínio, em outras palavras)".

Por exemplo, em determinada instituição pública o servidor pode, conforme o cargo ocupado, cumprir uma jornada de trabalho de 20, 30 ou 40 horas semanais. Considerando que Carga_Horária seja um dos atributos de Servidores, o conjunto formado pelos valores 20, 30 e 40 compõem o domínio dessa propriedade, uma vez que Carga_Horária tem, obrigatoriamente, que assumir um desses três valores.

No modelo E-R atributos são representados por elipses, conforme notação abaixo:

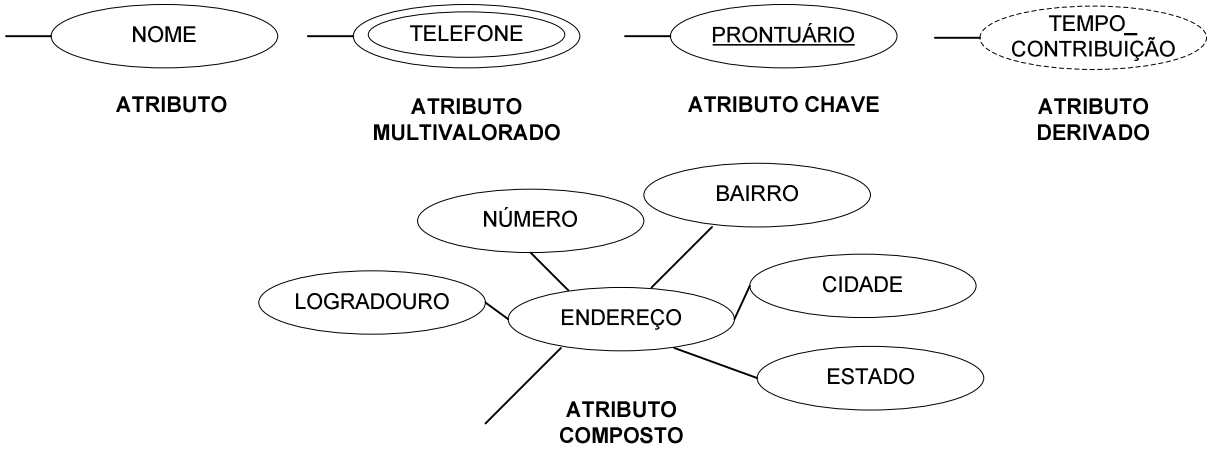


FIGURA 5 REPRESENTAÇÃO GRÁFICA DE ATRIBUTOS

6.3. RELACIONAMENTOS E CONJUNTOS DE RELACIONAMENTOS

"Um *relacionamento* é uma associação entre uma ou várias entidades" (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 24).

Ilustrando, quando nos referimos ao local onde cada servidor desempenha suas atividades – sua lotação – este dado não se refere somente à Servidores e nem unicamente à Campi, mas sim a ambos. Esse elemento dependente de uma e outra

entidade é representado no MER pelo que se denomina relacionamento. Então, em nosso exemplo, dizemos que os entes em Servidores estão associados aos entes em Campi através do relacionamento Lotação.

“Um *conjunto de relacionamentos* é um conjunto de relacionamentos do mesmo tipo (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 25). No modelo E-R conjuntos de relacionamentos são representados por losangos.

FIGURA 6 REPRESENTAÇÃO GRÁFICA DE RELACIONAMENTOS

Devemos ler o diagrama anterior da seguinte maneira: Servidores *têm como lotação* Campi, da esquerda para a direita; e Campi *é lotação de* Servidores, da direta para a esquerda.

“As entidades envolvidas em determinado relacionamento são ditas **participantes** desse relacionamento. O número de participantes em determinado relacionamento é chamado **grau** desse relacionamento” (DATE, 2004, p. 357).

Assim, no exemplo anterior tem-se um relacionamento de grau dois (também conhecido como relacionamento binário), pois há dois conjuntos de entidades participantes: Servidores e Campi.

Relacionamentos, da mesma forma que entidades, podem ter atributos descritivos (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 25). Por exemplo, suponhamos que os servidores possam, a interesse da administração pública, ser removidos de um pólo de trabalho a outro e que seja necessário armazenar em banco de dados o histórico das remoções efetuadas. Então, ao relacionamento Lotação, que associa Servidores e Campi, agrega-se o atributo Data_Início que representará a data de admissão do funcionário em um dado campus. Desta maneira, registraremos a movimentação do servidor em cada um dos campi nos quais ele venha a trabalhar.

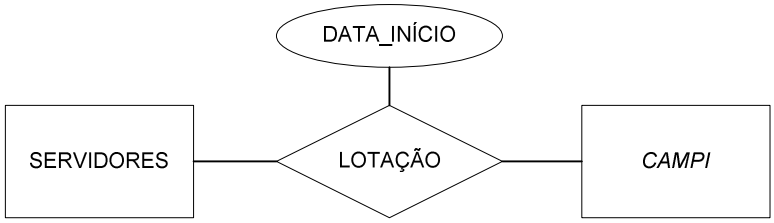


FIGURA 7 RELACIONAMENTO COM ATRIBUTO DESCRITIVO

6.4. AUTO-RELACIONAMENTOS

Relacionamentos entre entidades de mesma categoria denominam-se auto-relacionamentos (SETZER; CORRÊA DA SILVA, 2005, p. 47).

Mas, como compreender um relacionamento entre entidades de tipo?

Segundo ELMASRI e NAVATHE (2011, p. 141), “cada tipo de entidade que participa de um tipo de relacionamento desempenha nele uma função em particular”. SILBERSCHATZ, KORTH e SUDARSHAN (1999, p. 25), complementam dizendo que “a função que uma entidade desempenha em um relacionamento é chamada *papel*”. Então, em um auto-relacionamento (também denominado *relacionamento recursivo*) as entidades são do mesmo tipo, porém elas têm papéis diferentes.

Por exemplo, determinada autarquia federal precisa realizar concurso público para preencher vagas de emprego. Então, a autoridade máxima do órgão – o Reitor – designa uma comissão que será responsável por todos os trabalhos referentes ao processo seletivo. Essa comissão, por sua vez, pode delegar tarefas para grupos de trabalhos menores. Então, tem-se o seguinte cenário: uma comissão coordena comissões menores, sendo essas últimas subordinadas àquela primeira. Isso pode ser modelado através de um auto-relacionamento, onde o conjunto de entidades Comissão associa-se a ele mesmo através do relacionamento Supervisão, de maneira que Comissão ora desempenha o papel de supervisora, ora de supervisionada.

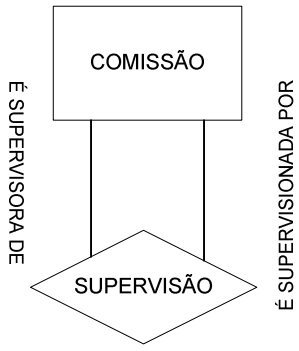


FIGURA 8 RELACIONAMENTO RECURSIVO

6.5. RESTRIÇÕES SOBRE TIPOS DE RELACIONAMENTO

De acordo com ELMASRI e NAVATHE (2011, p. 142), “os tipos de relacionamentos costumam ter certas restrições que limitam as combinações de entidades que podem participar no conjunto de relacionamentos correspondente”. Ainda segundo os autores, essas restrições que são estabelecidas de acordo com realidade que se é modelada, dividem-se em dois grupos: *razão de cardinalidade* e *participação*.

6.5.1. RAZÃO DE CARDINALIDADE OU CARDINALIDADE

Na definição de ELMASRI e NAVATHE (2011, p. 142), a razão de cardinalidade “especifica o número *máximo* de instâncias de relacionamento em que uma entidade pode participar”.

Em outras palavras, a cardinalidade “expressa o número de entidades às quais outra entidade pode estar associada via um conjunto de relacionamentos” (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 28). Conforme exemplificam esses autores, dado um conjunto de entidades A, um conjunto de entidades B e um conjunto de relacionamentos R (associação entre A e B), com relação à cardinalidade, tem-se uma das situações abaixo:

- UM PARA UM (1:1): cada entidade do conjunto de entidades A está associada a no máximo uma entidade do conjunto de entidades B, e cada entidade em B está associada a no máximo uma entidade em A.
- UM PARA MUITOS (1:N): uma entidade em A está associada a diversas entidades em B, mas cada entidade em B está associada a no máximo a uma entidade de A.
- MUITOS PARA MUITOS (M:N): uma entidade em A está associada a várias entidades em B, e uma entidade de B está associada a diversas entidades em A.

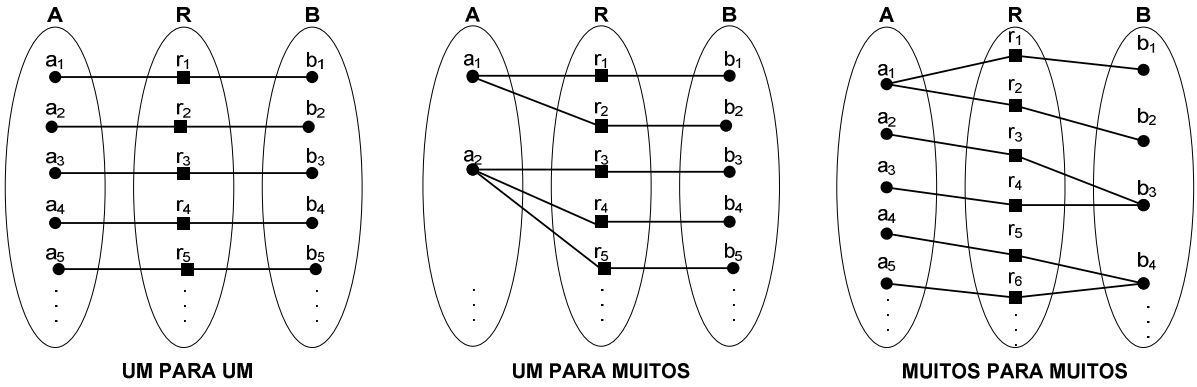


FIGURA 9 MAPEAMENTO DAS CARDINALIDADES
SILBERSCHATZ; KORTH; SUDARSHAN, 1999, P.29 (ADAPTADO)

6.5.2. RESTRIÇÕES DE PARTICIPAÇÃO E DEPENDÊNCIA DE EXISTÊNCIA

Para ELMASRI e NAVATHE (2011, p. 143), a restrição de participação “especifica o número *mínimo* de instâncias de relacionamento em que cada entidade pode participar”. Ainda segundo os autores, as restrições de participação classificam-se em *total* e *parcial*.

Considerando ainda os conjuntos de entidades A, entidades B e relacionamentos R, temos que a participação de A em R é dita *total* de todos os elementos em A participam de pelo menos um relacionamento R. Caso contrário, se apenas uma parte dos entes em A associam-se aos elementos em B através de R, a participação de A em R é chamada de *parcial*.

Outra questão importante, mas que está fortemente ligada à restrição de participação do tipo total, é a *dependência de existência*. Vamos explicá-la por meio do seguinte exemplo: se a participação de A em R é total, isto significa que a existência das entidades em A dependem da existência das entidades em B. Então, A é dito *dependente* de B.

No modelo E-R, a participação total (*no mínimo um*) é representada por linha dupla ligando o conjunto de entidades ao relacionamento correspondente. Já a participação parcial (*nenhum mínimo*) é exibida por linha simples.

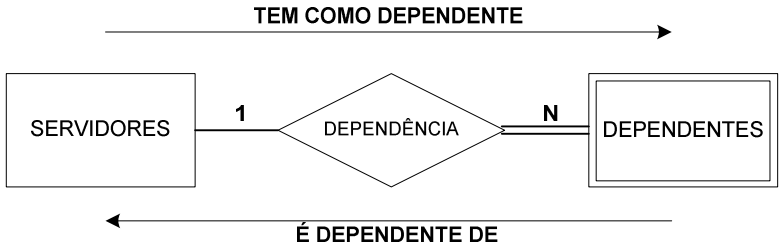


FIGURA 10 REPRESENTAÇÃO DE RESTRIÇÕES DE CARDINALIDADE E PARTICIPAÇÃO

O diagrama acima significa que um servidor tem como dependente no mínimo nenhum e no máximo muitos dependentes. Por outro lado, um dependente é dependente de no mínimo um e no máximo um servidor. Ou seja, poderão existir funcionários que não possuem dependentes, porém todo dependente deve estar associado a um servidor.

7. O MODELO ENTIDADE-RELACIONAMENTO ESTENDIDO (EER)

Os conceitos básicos do Modelo de Entidade-Relacionamento (E-R) são suficientes para construir o esquema de grande parte dos bancos de dados. Contudo, algumas situações são modeladas de maneira mais correta utilizando recursos adicionais ditos extensões do E-R. Essas ferramentas compreendem os conceitos de *super/subclasse*, *herança de atributos*, *especialização* e *generalização*. Todas elas, juntamente com os conceitos E-R vistos até aqui, compõem o denominado *Modelo de Entidade-Relacionamento Estendido (EER)*.

7.1. SUPERCLASSES, SUBCLASSES E HERANÇA DE ATRIBUTOS

“Um conjunto de entidades pode conter subgrupos de entidades que são, de alguma forma, diferentes de outras entidades do conjunto” (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 39).

Essa “diferença” consiste, por exemplo, em atributos que são específicos de determinado subtipo do conjuntos de entidades, ou ainda, relacionamentos que se aplicam somente a um dado subgrupo.

Por exemplo, o conjunto de entidades Servidores pode ser subdividido em Estatutários (servidores regidos pelo Regime Jurídico Único – RJU) e Celetistas (servidores regidos pela Consolidação das Leis Trabalhistas – CLT). Celetistas possuem o atributo CTPS (Carteira de Trabalho e Previdência Social) que é próprio dessa subcategoria. Estatuários possuem os atributos Portaria e Data_Publicação (número do ato legal de nomeação e sua respectiva data de publicação no Diário Oficial da União) que somente se aplicam a eles. Então, embora Estatutários e Celetistas pertençam ao conjunto de entidades Servidores, eles formam subcategorias com características peculiares.

Ainda, aproveitando o exemplo acima, diz-se que Celetistas e Estatutários são *subtipos* ou *subclasses* do tipo de entidades Servidores e o tipo de entidades Servidores é um *supertipo* ou *superclasse* para cada uma das subclasses.

Um ponto importante referente às subclasses é a *herança de atributos*. Segundo DATE (2004, p. 357) “as propriedades e os relacionamentos que se aplicam ao supertipo são **herdados** pelo subtipo”. Assim, todos os atributos e relacionamento de Servidores aplicam-se a Estatutários e Celetistas. Porém, vale salientar que o inverso não é verdadeiro – nem todos os atributos de Estatutários e Celetistas podem pertencer a Servidores.

7.2. ESPECIALIZAÇÕES E GENERALIZAÇÕES

“**Especialização** é o processo de definir um *conjunto de subclasses* de um tipo de entidade” (ELMASRI; NAVATHE, 2011, p. 163).

SILBERSCHATZ, KORTH e SUDARSHAN (1999, p. 40), esclarecem que uma especialização é representada no modelo EER por um triângulo com o rótulo ISA (do termo inglês “is a”, correspondente a “é um(a)”).

Utilizando o exemplo anterior, tem-se que *um Estatutário é um Servidor* e que *um Celetista é um Servidor*.

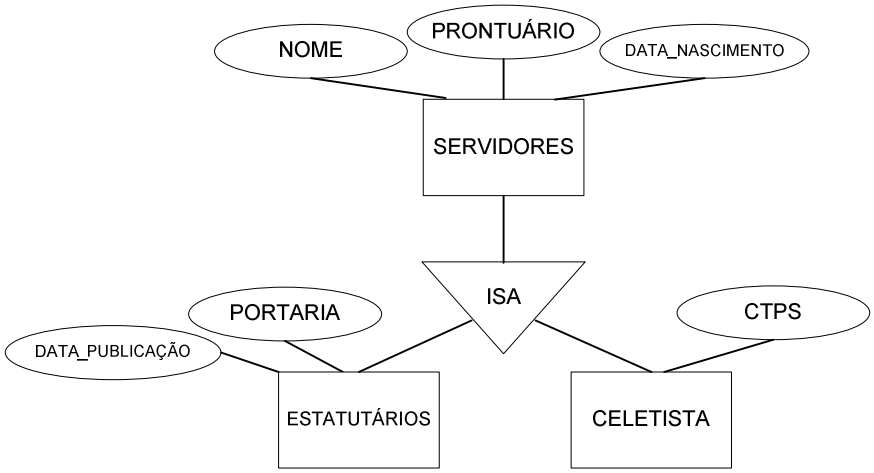


FIGURA 11 REPRESENTAÇÃO GRÁFICA DE ESPECIALIZAÇÃO

Os símbolos usados para representar as subclasses, seus atributos e relacionamentos são os mesmos que usamos na superclasse, ou seja, os mesmos vistos até aqui.

Generalização pode ser dita como o contrário da especialização. De acordo com ELMASRI e NAVATHE (2011, p. 164), neste processo nós “suprimimos as diferenças entre vários tipos de entidade, identificamos suas características comuns e as **generalizamos** em uma única **superclasse** da qual os tipos de entidade originais são **subclasses** especiais”.

Diz-se que a especialização é um *processo top-down* (do geral para o específico) enquanto a generalização é *bottom-up* (do específico para o geral) (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 41).

Em nosso exemplo, Servidores generaliza Celetistas e Estatutários.

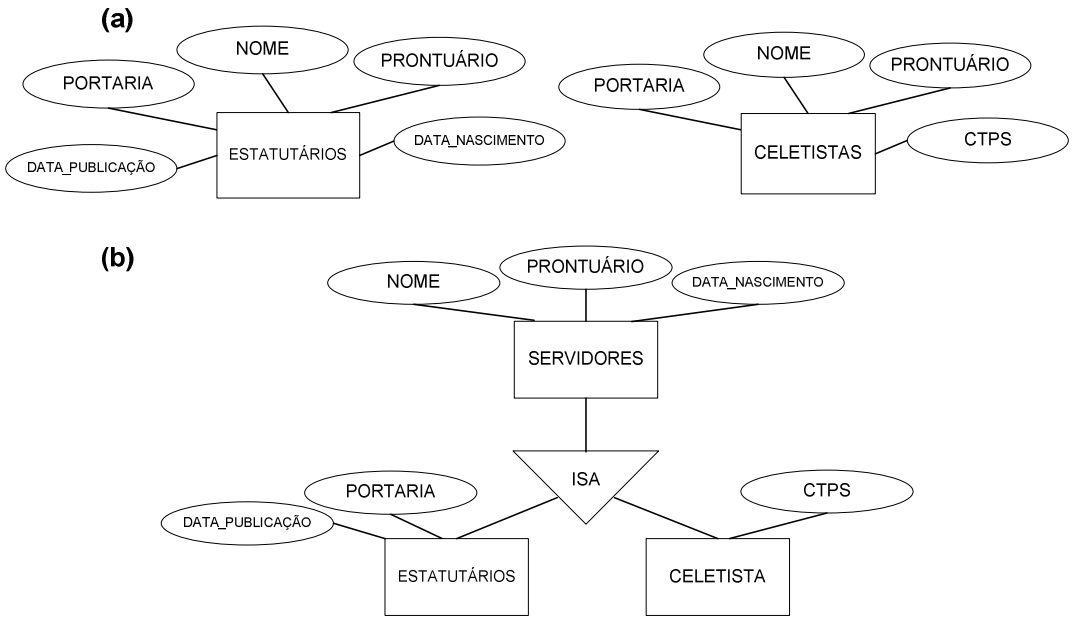


FIGURA 12 (a) DOIS TIPOS DE ENTIDADE, ESTATUTÁRIOS E CELETISTAS. (b) GENERALIZANDO ESTATUTÁRIOS E CELETISTAS NA SUPERCLASSE SERVIDORES.

7.2.1. RESTRIÇÕES DE ESPECIALIZAÇÕES E GENERALIZAÇÕES

Segundo ELMASRI e NAVATHE (2011, p. 164), algumas restrições podem ser aplicadas à especializações e generalizações. A primeira delas consiste em determinar quais entidades comporão cada uma das subclasses. Essa determinação dá-se pela escolha de um dos critérios a seguir:

- *Definição por condição ou predicado*: condiciona-se a admissão na subclasse ao valor que determinado atributo assume para uma dada entidade. Ou ainda, de acordo com o valor de um atributo da superclasse a entidade é alocada na

subclasse. Por exemplo, todas as entidades Servidores possuem o atributo Regime_Trabalho. Apenas aquelas entidades que atendam a condição Regime_Trabalho = “RJU” são incluídas na subclasse Estatutários. Somente as entidades que satisfaçam a condição Regime_Trabalho = “CLT” podem pertencer ao subtipo Celetista.

- *Definida pelo usuário*: quando não há como estabelecer uma condição para a escolha das entidades que estarão na subclasse. De acordo com os autores, nesses casos “a condição de membro é *especificada individualmente para cada entidade pelo usuário*, e não por qualquer condição que possa ser avaliada automaticamente”.

O segundo tipo de restrição determina se uma dada entidade pode pertencer a mais de uma subclasse. Assim, as subclasses podem ser:

- *Mutuamente exclusivas*: uma entidade da superclasse pode pertencer a no máximo uma subclasse. No exemplo utilizado, as subclasses são mutuamente exclusivas, porque cada entidade pertence a um, e somente um, subtipo.
- *Sobrepostas*: cada entidade da superclasse pode integrar uma ou mais subclasses.

A terceira restrição especifica se toda entidade da superclasse tem, obrigatoriamente, que pertencer a uma das subclasses. Essa restrição pode ser:

- *Total*: quando toda entidade da superclasse deve pertencer a uma subclasse.

A generalização Servidores é total, pois todas as entidades de Servidores integram os subtipos Celetistas e Estatutários.

- *Parcial*: uma entidade da superclasse pode (ou não) pertencer a uma subclasse.

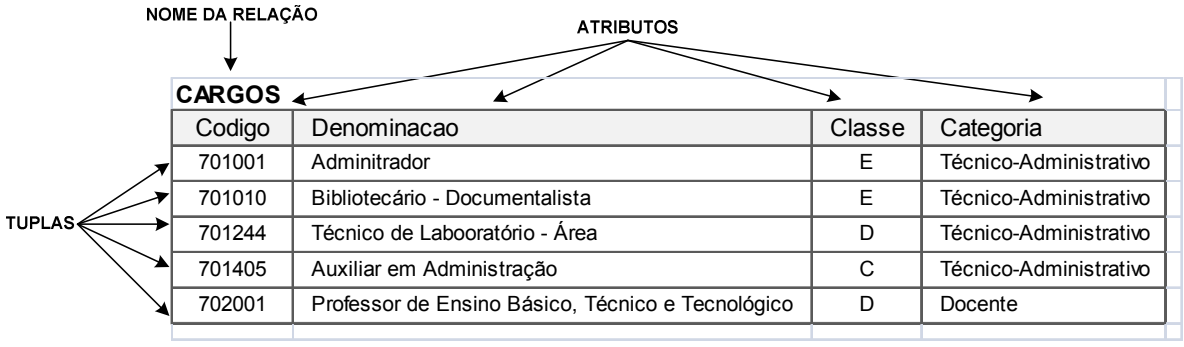
8. O MODELO RELACIONAL

O Modelo Relacional (MR) é um modelo de dados representativo (ou de implementação) que foi proposto por Ted Codd, em 1970. O modelo fundamenta-se em conceitos da matemática – teoria dos conjuntos e lógica de predicado. Os primeiros sistemas comerciais baseados no MR foram disponibilizados em 1980 e desde então ele vem sendo implementado em muitos sistemas, tais como Access, Oracle, MySql, entre outros (ELMASRI; NAVATHE, 2011, p. 38).

Para DATE (2004, p. 67), o modelo relacional refere-se a "três aspectos principais dos dados: a **estrutura** de dados, a **integridade** de dados e a **manipulação** de dados".

8.1. O ASPECTO ESTRUTURAL

No Modelo Relacional o banco de dados é representado como um conjunto de relações. Considerando que uma relação é, de certo modo, similar a uma tabela de valores e aplicando a terminologia do MR diz-se que as linhas denominam-se tuplas; as colunas, atributos; e a tabela em si, relação (ELMASRI; NAVATHE, 2011, p. 39).



Codigo	Denominacao	Classe	Categoria
701001	Adminitrador	E	Técnico-Administrativo
701010	Bibliotecário - Documentalista	E	Técnico-Administrativo
701244	Técnico de Labooratório - Área	D	Técnico-Administrativo
701405	Auxiliar em Administração	C	Técnico-Administrativo
702001	Professor de Ensino Básico, Técnico e Tecnológico	D	Docente

FIGURA 13 OS ATRIBUTOS E TUPLAS DE UMA RELAÇÃO *CARGOS*

O conjunto de valores que cada atributo pode assumir chama-se *domínio* (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 62).

Seja D_n, onde *D* é o domínio de determinado atributo e *n* a posição do atributo na relação e considerando a relação Cargos (figura 13), tem-se que D_1 é domínio do

atributo Codigo, pois representa o conjunto de todos os possíveis códigos de cargo. D_2 é o domínio de Denominacao, porque denota o conjunto de todas as nomenclaturas de cargo. D_3 é domínio de Classe, visto que representa o conjunto de todas as classes de cargo, e assim por diante.

Ainda segundo SILBERSCHATZ, KORTH e SUDARSHAN (1999, p. 62), uma relação é definida matematicamente como um subconjunto do produto cartesiano de uma lista de domínios. Desta forma, a relação Cargos é denotada por $D_1xD_2xD_3xD_4$, ou seja, Cargos é um subconjunto do conjunto de todas as combinações possíveis de valores.

Um *esquema de relação* (ou *scheme de relação*) é utilizado para descrever uma relação. Um esquema é indicado por $R(A_1, A_2,\ldots,A_n)$, onde R é o nome da relação e A_n são seus atributos (ELMASRI; NAVATHE, 2011, p. 40). Por exemplo, Cargos(Codigo, Denominacao, Classe, Categoria) representa o esquema da relação Cargos.

O número de atributos de um esquema de relação é denominado *grau* ou *aridade* (ELMASRI; NAVATHE, 2011, p. 40). Cargos, por exemplo, é uma relação de grau três.

Uma *instância de relação* (ou *estado de relação*) é um conjunto de tuplas – seus valores num dado momento. O estado *r* do esquema *R*, denotado por r(R), é um conjunto de tuplas $r=\{t_1, t_2,\ldots, t_n\}$, onde cada tupla *t* é formada por uma lista de valores $t=(v_1, v_2, \ldots, v_n)$, em que cada valor v_i está no domínio do respectivo atributo A_i (ELMASRI; NAVATHE, 2011, p. 40). Exemplificando, a instância do esquema da relação Cargos (figura 13) é composta de cinco tuplas, cada tupla é uma lista de quatro valores (por exemplo, na primeira tupla temos os seguintes valores: 701001, Administrador, E, Técnico-Administrativo), sendo cada valor elemento do domínio do atributo correspondente (701001 está no domínio do atributo Código, Administrador está no domínio de Denominacao, E está no domínio de Classe e Técnico-Administrativo está no domínio de Categoria).

8.2. O ASPECTO DE INTEGRIDADE

Discutiremos nesta seção os conceitos de: *superchave*, *chave*, *chave candidata*, *chave primaria*, *chave única* (ou *chave alternativa*), *chave estrangeira* e integridade *referencial*.

“Uma *superchave* é um conjunto de um ou mais atributos que, tomados coletivamente, nos permitem identificar de maneira unívoca uma entidade em um conjunto de entidades” (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 32).

Na relação Cargos (Figura 13), os conjuntos {Codigo, Categoria} e {Denominacao, Classe}, por exemplo, são superchaves. Porém, {Classe, Categoria} não é, visto que há para esse conjunto de atributos tuplas que tem as mesmas combinações de valores.

Na definição de DATE (2004, p. 233), um conjunto de atributos é dito *chave* se satisfazer as condições de:

1) *Unicidade*: esta é a propriedade atendida pelas superchaves, conforme especificado anteriormente. Ou seja, para um dado conjunto de atributos não há na relação tuplas com valores iguais.

2) *Irredutibilidade*: estabelece que não deve existir no conjunto de atributos chamado chave um subconjunto que tenha a propriedade de unicidade. Em outras palavras, além de seguir a primeira regra, a chave deve ser um conjunto mínimo de atributos.

Por exemplo, o conjunto {Codigo, Denominacao} não é uma chave, porque embora satisfaça a primeira regra acaba quebrando a segunda – {Codigo} e {Denominação}, tomados separadamente, já identificam exclusivamente cada tupla na relação Cargos. O conjunto {Codigo} é um exemplo de chave, pois atende simultaneamente as duas condições acima.

Uma relação pode possuir mais de uma chave, sendo cada uma delas chamada de *chave candidata* (ELMASRI; NAVATHE, 2011, p. 45). {Codigo} e {Denominacao}, por exemplo, são chaves candidatas.

A chave candidata usada para identificar tuplas em uma dada relação denomina-se *chave primária*. As demais chaves candidatas dessa relação são ditas *chaves únicas*. Indica-se uma chave primária sublinhando no esquema da relação os atributos que a compõem (ELMASRI; NAVATHE, 2011, p. 45).

Ilustrando, o esquema da relação Cargos ficaria assim: Cargos(<u>Codigo</u>, Denominacao, Classe, Categoria).

Tendo por base o exposto por SETZER e CORRÊA DA SILVA (2005, p. 124), discutiremos os conceitos de *chave estrangeira* e *integridade referencial* por meio de um exemplo.

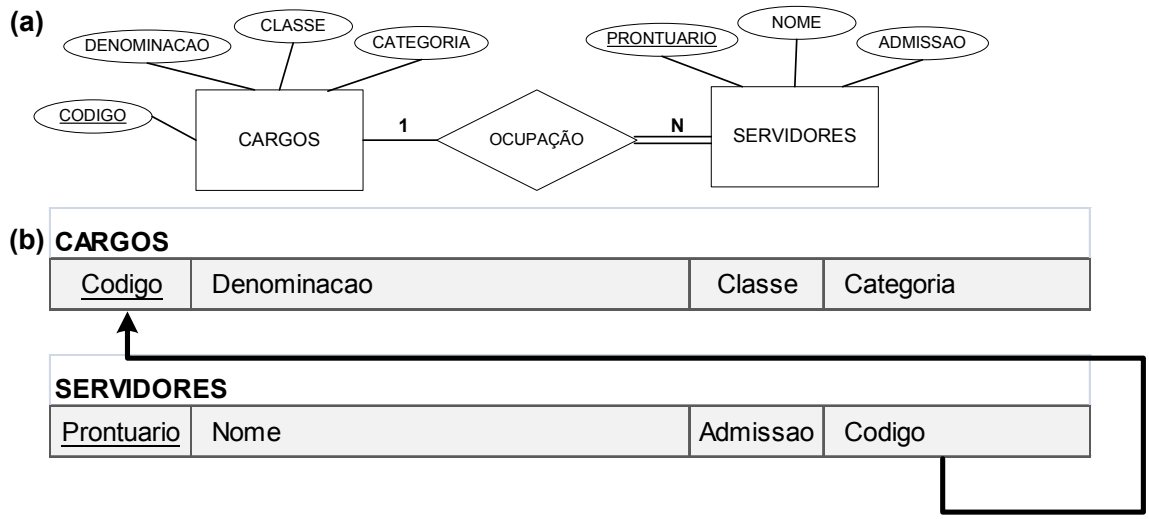


FIGURA 14 (a) DIAGRAMA ENTIDADE-RELACIONAMENTO. (b) RELAÇÕES *CARGOS* E *SERVIDORES*.

Considerando a representação acima, interpretamos do Diagrama Entidade-Relacionamento (figura 14a) que cada ente do conjunto de entidades Servidores ocupa no mínimo e no máximo um cargo. Entretanto, um cargo pode ser ocupado por nenhum ou muitos servidores. Mapeando essa situação para o Modelo Relacional, obtem-se os seguintes esquemas de relação: Cargos(Codigo, Denominacao, Classe, Categoria) e Servidores(Prontuario, Nome, Admissao, Codigo). Consequentemente, existirão duas relações conforme figura 14b.

Observe que o relacionamento *Ocupação* entre os conjuntos de entidades Servidores e Cargos, ocorre no Modelo Relacional (figura 14b) através da transposição do atributo *Codigo* de Cargos para Servidores. Esse atributo transposto em Servidores (tem como origem a relação Cargos e como destino a relação Servidores) denomina-se *chave estrangeira*. Uma tupla em Servidores faz referência a uma tupla de Cargos, sendo que essa referência é realizada através do valor contido no atributo Codigo.

Chama-se *integridade referencial* a regra de que o valor contido na chave estrangeira de Servidores deve corresponder a um valor de chave primária em Cargos. Em outras palavras, a integridade referencial é a restrição de que “o banco de dados não pode conter quaisquer valores de chaves estrangeiras não correspondentes” (DATE, 2004, p. 237).

8.3. O ASPECTO MANIPULATIVO

"Uma *linguagem de consulta* é a linguagem por meio da qual os usuários obtêm informações do banco de dados" (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 68).

De acordo com SETZER e CORRÊA DA SILVA (2005, p. 172), foram definidas duas linguagens de acesso ao modelo relacional: a *Álgebra Relacional* e o *Cálculo Relacional*.

Segundo SILBERSCHATZ, KORTH e SUDARSHAN (1999, p. 68), as linguagens de consulta podem ser classificadas em *procedurais* ou *não-procedurais,* sendo a Álgebra Relacional um exemplar da primeira modalidade e o Cálculo Relacional um exemplo da segunda.

Em uma linguagem de consulta procedural é necessário especificar todas as operações a serem realizados sobre o banco de dados para a obtenção da informação desejada. Ao contrário, em uma linguagem não-procedural, descreve-se a informação desejada sem especificar o procedimento para obtê-la.

A abordagem das linguagens de Álgebra e Cálculo Relacional está fora do escopo deste trabalho, porém, para fixar a definição dada acima utilizaremos o seguinte exemplo: considerando o esquema da relação Cargos, digamos que seja necessário saber a denominação e a categoria dos cargos cuja classe seja "C".

Para cumprir a tarefa proposta utilizando a Álgebra Relacional, devemos saber quais operadores usar e em que ordem eles serão aplicados. Obrigatoriamente, temos que saber: a) que serão necessários dois operadores (o de *Seleção*, indicado por σ (sigma), e o operador *Projeção*, indicado por π (pi)) e b) que primeiro deve ser executada a seleção, e depois a projeção. Assim, a expressão final teria a seguinte forma:

$$\pi_{\text{Denominação, Categoria}}(\sigma_{\text{Classe = "C"}}{}^{(\text{Cargos})})$$

Já utilizando o Cálculo Relacional, é necessário especificar apenas o que desejamos. Desta forma, especificarmos quais os atributos desejados para cada tupla, informamos que essas tuplas estão na relação *Cargos* e que devem ser

listadas somente aquelas cujo valor do atributo Classe seja “C”. A instrução em Cálculo Relacional teria a seguinte sintaxe:

{ t.Denominacao, t.Categoria | Cargos(t) *and* t.Classe = “C”}

9. O PADRÃO-SQL

A Structured Query Language (SQL) ou Linguagem de Consulta Estruturada foi criada pela IBM Research, no início da década de 1970, para o protótipo de um sistema de banco de dados chamado System R (DATE, 2004, p. 71).

Baseada nas linguagens de Álgebra e Cálculo Relacional, e inicialmente denominada SEQUEL (Structured English QUEry Language), SQL hoje é a linguagem padrão para Sistemas Gerenciadores de Bancos de Dados Relacionais (SGBDR), sendo mais intelegível do que suas linguagens maternas – consideradas técnicas demais para o usuário (ELMASRI; NAVATHE, 2011, p. 57).

A SQL é padronizada pelo American National Standards Institute (ANSI) e pela International Standards Organization (ISO), conjuntamente. A primeira versão-padrão, chamada de SQL-86 (ou SQL1), foi lançada em 1986 e desde então ela vem sendo atualizada – SQL-92 (ou SQL2), SQL:1999 (ou SQL3), SQL:2003, SQL:2006 e, ainda, uma outra atualização ocorreu em 2008 (ELMASRI; NAVATHE, 2011, p. 57).

Apesar de conhecida como uma “linguagem de consulta”, a SQL oferece também recursos para definir a estrutura dos dados, atualizar – incluir, excluir e alterar – dados, especificar restrições de integridade e outros recursos mais (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 109).

De acordo com DATE (2004, p. 85), a SQL possui, entre outros, os seguintes componentes: *Data Definition Language* (DDL) ou *Linguagem de Definição de Dados* e *Data Manipulation Language* (DML) ou *Linguagem de Manipulação de Dados*.

9.1. LINGUAGEM DE DEFINIÇÃO DE DADOS

A DDL é usada para especificar relações, domínios, regras de integridade, entre outros. O comando *create*, que na visão de ELMASRI e NAVATHE (2011, p. 58) é a

principal instrução para a definição de dados, é utilizado para criar tabelas, assertions, domínios, triggers e mais. A DDL também inclui as intruções *alter* e *drop*.

9.1.1. INSTRUÇÃO CREATE TABLE

Os termos *relação*, *tupla* e *atributo* do Modelo Relacional correspondem à *tabela*, *linha* e *coluna*, respectivamente, na SQL (DATE, 2004, p. 72).

A instrução *create table* é usada para criar uma *tabela*. Na visão de SETZER e CORRÊA DA SILVA (2005, p. 174), "a declaração de uma tabela resume-se à descrição do seu esquema, dos domínios, de integridades referenciais e certas restrições de integridade"

Antes de seguirmos adiante, discutiremos os conceitos de *tipo de dado* (domínio) e *restrições* (de atributo, chave e integridade referencial).

9.1.1.1. Domínios e tipos de dados

"Um método comum de especificação de um domínio é definir um tipo de dado do qual são retirado os valores de dados que formam o domínio" (ELMASRI; NAVATHE, 2011, p. 39).

Os tipos de dados podem ser *definidos pelo sistema* (também conhecidos como *internos* ou *embutidos*) ou *definidos pelo usuário* (DATE, 2004, p. 96).

De acordo com SILBERSCHATZ, KORTH e SUDARSHAN (1999, p. 138), os principais tipos disponibilizados pelo padrão SQL (tipos embutidos) são:

- *char(n)*: para sequência de caracteres de tamanho fixo, em que o tamanho *n* é definido pelo usuário;
- *varchar(n)*: para sequência de caracteres de tamanho variável, em que o tamanho máximo *n* é definido pelo usuário;
- *int*: para números inteiros;
- *smallint*: para números inteiros pequenos (é um subconjunto de int);

- *numeric(p,d)*: para números de ponto fixo, em que *p* é a quantidade total de dígitos que compõe o número e *d* especifica quantos desses dígitos estão à direita (após o ponto decimal). Um numeric(5,2) permite armazenar um número com cinco dígitos, sendo que dois deles estão após o ponto decimal, ou seja, o número armazenado deve possuir o seguinte formato 999.99.
- *real* e *double precision*: para números de ponto flutuante e ponto flutuante de precisão dupla, em que a precisão depende do equipamento utilizado.
- *float(n)*: para números de ponto flutuante, em que a precisão *n* é definida pelo usuário. Assim, float(2) armazena números com duas casas decimais.
- *date*: para datas, inclui ano, mês e dia; e
- *time*: para horários, inclui hora, minuto e segundo.

"Um domínio pode ser declarado e seu nome usado com a especificação do atributo" (ELMASRI; NAVATHE, 2011, p. 61). São os tipos de dados *definidos pelo usuário.* Para isso usamos a instrução *create domain*, conforme abaixo:

create domain <nome do domínio> *as* <tipo de dado>

Por exemplo, vamos declarar o domínio *Moeda* e utilizá-lo mais adiante na especificação do atributo *Salario*.

create domain Moeda *as numeric(9,2)*

9.1.1.2. Restrições de atributo, chave e integridade referencial

De acordo com ELMASRI e NAVATHE (2011, p. 61), podemos especificar restrições de atributo, chave e integridade referencial como parte da criação de tabelas através do acréscimo de algumas cláusulas:

- *not null:* usada para especificar atributos cujo valor não possa ser vazio, ou seja, estabelece que a coluna da tabela não conterá valores nulos.
- *default*: define um valor padrão para dado atributo. Deste modo, ao se criar uma nova linha na tabela, esse valor será automaticamente atribuído à coluna caso nenhum valor seja explicitamente informado. Se a cláusula default for omitida, o valor padrão é *null* para aqueles atributos que não contenham a restrição *not null*.
- *check*: empregada para limitar os valores de um atributo. Assim, toda linha da tabela deve satisfazer a condição especificada na cláusula *check*.

- *primary key*: usada para especificar a chave primária de uma relação.
- *unique*: aplicada na especificação da chave única de uma relação.
- *foreign key*: utilizada para especificar a chave estrangeira de uma relação. Essa cláusula também impõe integridade referencial ao banco de dados.
- *constraint*: opcionalmente utilizada para nomear uma restrição. Restrições com um nome são passíveis de exclusão ou substituição, caso necessário.

As três primeiras cláusulas – *not null, default* e *check* – são ditas *restrições de atributo*. *Primary key* e *Unique* são *restrições de chave*. *Foreign key* é uma *restrição de integridade referencial*. A sintaxe de cada cláusula será exposta por meio de exemplo, conforme veremos adiante.

Retomando o ponto onde paramos, segundo SILBERSCHATZ, KORTH e SUDARSHAN (1999, p. 139), a instrução *create table* possui a seguinte forma:

create table r (A_1D_1, A_2D_2, …, A_nD_n,
<regras de integridade$_1$>,
…,
<regras de integridade$_k$>),

onde *r* é o nome da relação, A_i é o nome do atributo no esquema da relação *r* e D_i é o domínio do atributo A_i. As regras de integridade incluem restrições de atributo, chave e integridade referencial.

Para exemplificar o comando *create table*, considere os seguintes esquemas de relação:

Cargos(Codigo, Denominacao, Classe, Categoria)

Campi(Sigla, Cidade)

Servidores(Prontuario, SIAPE, Nome, Admissao, Salario, CodCargos, CodCampi)

Aplicando os conceitos discutidos, podemos especificar esses esquemas através das instruções SQL abaixo:

create table Cargos
(Codigo *char*(6) *not null*,
Denominacao *varchar*(50) *not null*,
Classe *char*(1) *default* ‘E’,
Categoria *varchar*(22) *check*(Categoria *in* (‘Tecnico-Administrativo’, ‘Docente’),
constraint ChPCargos *primary key* (Codigo))

create table Campi
(Sigla *char*(8) *not null*,
Cidade *varchar*(25),
constraint ChPCampi *primary key* (Sigla))

create table Servidores
(Prontuario *char*(6) *not null*,
SIAPE *char*(7) *not null*,
Nome *varchar*(70),
Admissao *date*,
Salario *Moeda*,
CodCargos *char*(6) *not null*,
CodCampi *char*(6) *not null*,
constraint ChPServidores *primary key* (Prontuario),
constraint ChAServidores *unique* (SIAPE),
constraint ChEServCargo *foreign key* (CodCargos) *references* Cargos (Codigo),
constraint ChEServCampi *foreign key* (CodCampi) *references* Campi (Sigla))

Aqui, criamos uma tabela denominada *Cargos* que possui quatro colunas. Dissemos que a segunda coluna (Denominacao) não aceita valores nulos. A primeira coluna (Codigo) é a chave primária (primary key) da tabela Cargos e por esse motivo também não permite valores nulos. Especificamos que o valor padrão (default) para *Classe* é “E”. Através da cláusula check determinamos que *Categoria* deve assumir o valor “Técnico-Administrativo” ou “Docente”. E, ainda, definimos uma restrição (constraint) de chave primária chamada *ChPCargos*.

Criamos também uma tabela *Campi* com duas colunas – *Sigla* (que é chave primária) e *Cidade* – e também uma restrição de chave primária denominada *ChPCampi*.

A última tabela (Servidores) é composta por sete colunas, sendo: *Prontuario*, uma chave primária; *SIAPE*, uma chave única (ou alternativa); *CodCargos*, uma chave estrangeira que referencia a tabela *Cargos*; *CodCampi*; uma chave estrangeira que referencia a tabela *Campi*; e mais três colunas, *Nome*, *Admissao* e *Salario* (note que para especificar esse atributo usamos o nome do domínio que foi definido anteriormente). Além disso, existem quatro restrições: duas de chave estrangeira

(ChEServCargo e ChEServCampi), uma de chave única (ChAServidores) e outra de chave primária (ChPServidores).

Abordaremos agora outros dois importantes comandos pertencentes à DDL – *drop* e *alter.*

9.1.2. INSTRUÇÃO DROP

"O comando DROP pode ser usado para remover elementos *nomeados* do esquema, como tabelas, domínios ou restrições" (ELMASRI; NAVATHE, 2011, p. 91). Ele possui a seguinte sintaxe:

drop <tipo do elemento> <nome do elemento>

Suponha, por exemplo, a necessidade de excluir a tabela *Servidores,* assim como o domínio *Moeda*. As respectivas instruções seriam definidas assim:

drop table Servidores

drop domain Moeda

9.1.3. INSTRUÇÃO ALTER

A instrução *alter* permite modificar a estrutura de uma tabela, bem como alterar a definição de outros elementos nomeados de um esquema. Considerando uma tabela, tem-se a oportunidade de adicionar e excluir colunas, modificar a definição de colunas, e ainda, acrescentar ou eliminar restrições para a tabela (ELMASRI; NAVATHE, 2011, p. 61).

Entre outras, podemos utilizar o comando *alter* das seguintes formas:

- *alter table* <nome da tabela> *add column* <nome da coluna> <tipo de dado>, para acrescentar uma coluna a uma tabela;
- *alter table* <nome da tabela> *drop column* <nome da coluna>, para remover uma coluna de uma tabela;
- *alter table* <nome da tabela> *add constraint* <nome da restrição>, para adicionar uma restrição a uma tabela; e

- *alter table* <nome da tabela> *drop constraint* <nome da restrição>, para excluir uma restrição de uma tabela.

9.2. LINGUAGEM DE MANIPULAÇÃO DE DADOS

Uma vez definido o banco de dados é possível operar sobre ele através das operações de manipulação: *select*, *insert*, *update* e *delete* (DATE, 2004, p. 73).
Select é usada para realizar consultas ao banco de dados, enquanto *insert, update* e *delete* são aplicadas na inserção, atualização e exclusão de dados, respectivamente.

9.2.1. INSTRUÇÃO SELECT

De acordo com ELMASRI e NAVATHE (2011, p. 86), um comando *select* pode incluir até seis cláusulas:

select <lista de atributos e funções>
from <lista de tabelas>
where <condição>
group by <atributo de agrupamento>
having <condição de grupo>
order by <lista de atributos>,

devendo a instrução seguir essa ordem e sendo obrigatórias apenas as duas primeiras cláusulas – *select* e *from*.

9.2.1.1. Cláusulas select e from

A cláusula *select* é usada para elencar os atributos desejados no resultado da consulta, e *from,* para especificar em qual ou quais tabelas eles são encontrados.

Considerando a tabela *Cargos*, podemos, por exemplo, criar a instrução:

select Codigo, Denominacao
from Cargos

para listar somente as colunas (atributos) *Codigo* e *Denominacao* da tabela *Cargos*. Agora considere a seguinte consulta: “apresente todas as classes da tabela cargos”.

select Classe
from Cargos

O resultado da consulta é uma tabela com apenas uma coluna onde aparecem os valores de classe. Contudo, alguns desses valores podem aparecer duplicados caso tenhamos na tabela dois ou mais cargos que possuam a mesma classe (o que é bem provável). Para forçar a eliminação dos valores duplicados no resultado dessa consulta devemos acrescentar a palavra reservada *distinct*.

select distinct Classe
from Cargos

Assim, asseguramos que cada valor “D”, por exemplo, apareça uma única vez no resultado da consulta. Entretanto, se desejarmos que de fato apareçam todos os valores podemos declará-lo de forma explícita por meio da palavra-chave *all*.

select all Classe
from Cargos

O comando *select* sem *distinct ou all*, todavia, é correspondente a *select all (*ELMASRI; NAVATHE, 2011, p. 68)

Caso seja necessário listar na consulta todas as colunas de uma tabela, podemos utilizar o sinal de asterisco (*****) ao invés de especificar os atributos na cláusula *select.* Desse modo,

select *******
from Cargos

é equivalente a

select Codigo, Denominacao, Classe, Categoria
from Cargos.

A cláusula *select* pode ainda conter expressões aritméticas envolvendo constantes ou atributos e operadores de +, -, * e / (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 112). Considerando a tabela *Servidores*, a instrução

select Nome, Salario * 12
from Servidores

lista o nome e o salário bruto anual de cada servidor, sendo essa última coluna obtida pela multiplicação do salário percebido pela quantidade de meses no ano.

Tabelas (relações) e colunas (atributos) podem ser renomeadas através da cláusula *as* (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 114). Tomemos como ilustração o resultado da consulta realizada acima. Nela a coluna onde aparecerá o salário bruto anual não terá um nome. Então, podemos atribuí-lo por meio da cláusula *as,* conforme segue:

select Nome, Salario * 12 *as* Sal_Bruto_Anual
from Servidores

9.2.1.2. Cláusula where

A cláusula *where* é denotada por *where*<*condição*>, onde "<condição> é uma expressão condicional (booleana) que identifica as tuplas a serem recuperadas pela consulta" *(*ELMASRI; NAVATHE, 2011, p. 64). Em outras palavras, todas as tuplas resultantes de uma consulta onde *where* esteja presente devem satisfazer a condição especificada nessa cláusula.

Por exemplo, a consulta "encontre o nome de todos os servidores que percebem salário acima de R$1.000,00" é especificada em SQL da seguinte forma:

select Nome
from Servidores
where Salario > 1000

Assim, a consulta seleciona em *Servidores* as linhas que atendam a condição de *where* (Salario>1000) e então projeta o resultado no atributo *Nome*.

Para montar condições na cláusula *where* é possível utilizar operadores de comparação (>, <, =, >=, <=, <> e between) e conectores lógicos (and, or e not) (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 113). Para ilustrar, veja como fica a sintaxe da consulta "selecione o prontuário e o nome dos servidores que foram admitidos entre 02/04/2010 e 07/09/2011".

select Prontuario, Nome
from Servidores
where Admissao *between* **'**02/04/2010' *and* '07/09/2011'

Na cláusula *where* é possível estabelecer condições que envolvam a comparação de substrings por meio do operador *like* associado ao caracter reservado *%* (porcentagem) ou _ (sublinhado), onde % representa uma cadeia de caracteres e _ denota um único caracter (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 115).

Por exemplo, suponhamos que deva ser montada uma lista de todos os servidores cujos nomes iniciam-se com a letra “A”. Para atender a essa solicitação seria necessária a seguinte consulta:

select Nome
from Servidores
where Nome *like* ‘A%’

Consideremos agora uma consulta que envolva colunas de duas tabelas, como “para os cargos que possuem ao menos um servidor, relacione o nome do servidor e o cargo por ele ocupado”.

De acordo com SETZER e CORRÊA DA SILVA (2005, p. 179), “para fazer uma consulta envolvendo duas tabelas ligadas, é sempre necessário colocar no predicado a condição de junção. Essa condição é justamente a igualdade das duas colunas”.

Ora, a chave primária de *Cargos* (Codigo) foi transposta em *Servidores* (CodCargos). Logo, CodCargos e Codigo são correspondentes. Assim, para solucionar a questão anterior teríamos a consulta:

select Nome, Denominacao
from Servidores, Cargos
where CodCargos **=** Codigo

9.2.1.3. Cláusulas group by

Antes de tratarmos da cláusula *group by* discutiremos *funções agregadas*.

São ditas *funções agregadas* as funções que recebem como entrada um conjuntos de valores, produzindo como saída um único valor. Compreendem as funções *avg*, *min*, *max*, *sum* e *count*, empregadas no cálculo de média, mínimo, máximo, totalização de valores e totalização de quantidades, respectivamente (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 120).

Suponha a consulta “encontre a quantidade total de servidores da instituição”. Para obter esse quantitativo podemos, por exemplo, totalizar a quantidade de prontuários existentes na tabela *Servidores*, conforme segue:

select count (Prontuario)
from Servidores

Essa consulta tem como resultado uma tabela formada por uma única linha e coluna onde consta a quantidade total de servidores do órgão.

Agora, sofisticando um pouco a consulta anterior, imagine que devamos “encontrar a quantidade total de servidores em cada campus da instituição”. Primeiro é necessário dividir os servidores em grupos de acordo com o seu local de trabalho e depois efetuar a contagem dos respectivos prontuários por meio de *count*.

A cláusula SQL *group by* é utilizada justamente quando precisamos aplicar funções agregradas à subgrupos do total de linhas, sendo que esses grupos serão formados a partir dos atributos constantes na cláusula group by (SILBERSCHATZ; KORTH; SUDARSHAN, 1999, p. 120). Enquanto “as **funções de agregação** são usadas para resumir informações de várias tuplas em uma síntese de tupla única. O **agrupamento** é usado para criar subgrupos de tuplas antes do resumo” *(*ELMASRI; NAVATHE, 2011, p. 82).

A instrução correspondente ao exemplo adotado ficaria assim:

select Denominacao, *count* (Prontuario)
from Servidores, Campi
where CodCampi = Sigla
group by Denominacao

9.2.1.4. Cláusula having

Imagine que para a consulta anterior seja necessário “encontrar apenas os *campi* que possuem trinta ou mais servidores”. Para atender essa demanda empregaremos a cláusula *having*.

Uma cláusula *having* possibilita selecionar somente grupos cujos componentes atendam a uma determinada condição (SETZER; CORRÊA DA SILVA, 2005, p. 184). No entanto, é importante ressaltar que *having* e *where* desempenham funções

diferentes. “A cláusula HAVING está para grupos como a cláusula WHERE esta para linhas; em outras palavras, HAVING é usada para eliminar grupos, da mesma maneira que WHERE é usada para eliminar linhas” (DATE, 2004, p. 204).

A tabela resultante pode ser obtida da seguinte forma:

```
select  Denominacao, count (Prontuario)
from Servidores, Campi
where CodCampi = Sigla
group by Denominacao
having count (Prontuario) >= 30
```

9.2.1.5. Cláusula order by

O último possível componente de um comando *select* é *order by*. Uma cláusula *order by* é usada para indicar que se deseja um ordenamento – ascendente (asc) ou descendente (desc) – para as linhas que compõe a tabela resultante, sendo a classificação ascendente o padrão, caso não seja explicitamente especificada uma ordem.

Para a ordenação descendente da coluna *Denominacao* no resultado da consulta anterior, teríamos:

```
select  Denominacao, count (Prontuario)
from Servidores, Campi
where CodCampi = Sigla
group by Denominacao
having count (Prontuario) >= 30
order by Denominacao desc
```

SETZER e CORRÊA DA SILVA (2005, p. 185), sintetizam de forma muito clara o funcionamento de comando *select* (com todas as suas possíveis cláusulas), da seguinte forma:

> Conceitualmente, a execução de cada **select** segue a seguinte ordem. 1. A cláusula **where** é testada produzindo com isso junção e/ou seleção de linhas. 2. É feito o agrupamento das linhas resultantes usando-se os valores das colunas do **group by**. 3. São escolhidos apenas os grupos que satisfazem a cláusula **having**, que é sempre aplicada a cada grupo como um todo e não individualmente

> às suas linhas. 4. As linhas assim resultantes são ordenadas pelas colunas indicadas no **order by**. 5. É feita a projeção na lista de colunas do **select**, eventualmente com cálculo de funções de agregação que são aplicadas a todas as linhas resultantes de cada grupo.

9.2.2. INSTRUÇÃO INSERT

O comando *insert* é usado para inserir dados em uma tabela, mais especificamente inserir linhas na tabela.

No comando de inserção podemos informar somente os valores a serem incluídos, porém devemos fazê-lo na mesma ordem em que as colunas (atributos) aparecem na tabela. Assim, para acrescentar novos dados à tabela *Cargos* temos que apresentar valores, obrigatoriamente, na seguinte ordem: em primeiro lugar o código do cargo; em segundo, sua nomenclatura; em terceiro, a classe a qual ele pertence; e por último, a categoria na qual ele se enquadra.

insert into Cargos

values (“701711”, “Auxiliar em Enfermagem”, “C”, “Tecnico-Administrativo”)

Entretanto, também é possível determinar a ordem em que os valores serão acréscidos à tabela especificando seus respectivos atributos na instrução *insert*. Considerando a tabela *Cargos*, para inclusão dos mesmos valores acima informados, mas em uma ordem diferente, escreveríamos:

insert into Cargos (Denominacao, Categoria, Codigo, Classe)

values (“Auxiliar em Enfermagem”, “Tecnico-Administrativo”, “701711”, “C”)

9.2.3. INSTRUÇÃO DELETE

A instrução *delete* é usada para eliminar linhas em tabela. O comando possui a seguinte sintaxe:

delete from <nome da tabela>

where <condição>,

onde <nome da tabela> é a tabela de onde serão excluídas as linhas e <condição> é um “filtro” que selecionará as linhas que serão eliminadas.

Por exemplo, imagine que devamos excluir da tabela *Cargos* todos os cargos cuja classe seja “A”. Isso pode ser realizado com:

delete from Cargos
where Classe = “A”

9.2.4. INSTRUÇÃO UPDATE

O comando *update* é usado para alterar o valor de determinado atributo de uma ou várias linhas em uma tabela.

Para ilustrar, considere que seja necessário conceder um aumento de 10% a todos os servidores. Faríamos isso através da instrução *update* abaixo:

update Servidores
set Salario = Salario + Salario * 0,01

Agora, suponha que o aumento deva ser concedido apenas aos servidores que percebem salário inferior a R$ 1.500, 00. Isso pode ser feito com:

update Servidores
set Salario = Salario + Salario * 0,01
where Salario < 1500

Encerramos aqui nossa breve explanação sobre a SQL. Não abordamos todos os tópicos pertencentes à linguagem, haja vista que é propósito deste trabalho apenas introduzir o assunto.

10. ESTUDO DE CASO: MICROSOFT OFFICE ACCESS E O IFSP

O Microsoft Office Access, ou denominado simplesmente Access, é um “aplicativo” voltado para a criação e a manipulação de bancos de dados relacionais. Ele compõe a suíte de programas do pacote *Office* que atualmente está em sua versão 2010.

Na definição acima usamos o termo “aplicativo”, pois na literatura existe uma gama de classificações distintas. Alguns o consideram um Sistema Gerenciador de Banco de Dados; outros, porém, chegam a dizer que não se trata nem de um banco de dados.

Contudo, para uma conceituação mais formal, compartilhamos da visão de CARVALHO (2006, p. 451), cujo entendimento é de que “o Microsoft Access é um SGBDR de pequeno porte para ser usado com bancos de dados pessoais e de pequenas e médias empresas”.

De acordo com OLIVEIRA (2005, p. 6), embora o Access tenha algumas limitações, ainda assim ele adequado para tratar uma quantidade razoável dados, porque “atende aos pequenos sistemas e prepara os dados para um futuro *up*”. Ou seja, ele é uma possível solução para a necessidade organizar dados e ainda é um bom precedente à adoção de sistemas gerenciadores de bancos de dados relacionais mais robustos como, por exemplo, o SQL SERVER.

SETZER e CORRÊA DA SILVA (2005, p. 215), em complemento, dizem que o Access apresenta as seguintes vantagens:

- é um sistema amplamente conhecido e que possui compatibilidade com suas versões anteriores;
- é de fácil utilização e possui interface amigável; e
- não requer conhecimentos profundos sobre a teoria bancos de dados.

Entretanto, cabe aqui a nota de que “não requerer informação aprofundada sobre banco de dados” não significa “não ter nenhum conhecimento sobre a área”. Aliás, para OLIVEIRA (2005, p. 3), esse é um dos motivos pelo qual o Access é temido, porque “exige do usuário o conhecimento de uma série de conceitos como registros, campos, relacionamentos, banco de dados, entre outros”.

Desta forma, para criar e manter de forma otimizada um banco de dados Access é muito recomendado o entendimento e a aplicação de todos os conceitos vistos nos capítulos anteriores: Modelo Entidade-Relacionamento, para a modelagem do banco de dados; Modelo Relacional, para o mapeamento do Diagrama Entidade-Relacionamento; e SQL, para a construção e manutenção da base de dados.

10.1. UMA VISÃO GERAL DO ACCESS

Um banco de dados Access é constituído por alguns elementos chamados *objetos*. Os objetos que compõe um banco Access são *tabelas*, *consultas*, *formulários*, *relatórios*, *macros* e *módulos.*

10.1.1. TABELAS

"Como em qualquer modelo relacional, a estrutura de dados básica do Access consiste em tabelas" (SETZER; CORRÊA DA SILVA, 2005, p. 219).

Tabelas são elementos formados por colunas e linhas, onde cada coluna corresponde a um campo, e cada linha, a um registro.

"Uma coluna é definida pelo seu nome, pelo tipo de dados que pode receber e por um comentário. O comentário é opcional" (SETZER; CORRÊA DA SILVA, 2005, p. 220).

Uma tabela pode ser criada de algumas maneiras, entre elas:

- no *Modo Design*, usando interface gráfica, conforme figura 15. O item 1 é o nome da tabela; 2 é onde especificamos o nome da coluna; em 3 definimos o tipo de dado que será armazenado; em 4 colocamos, opcionalmente, uma descrição do conteúdo que a coluna irá armazenar; 5 é onde definimos as propriedade do campo, como: tamanho, valor padrão (default), requerido (se aceita valores nulos ou não), entre outros; e 6 apresenta indicativo de que o campo é uma chave primária.

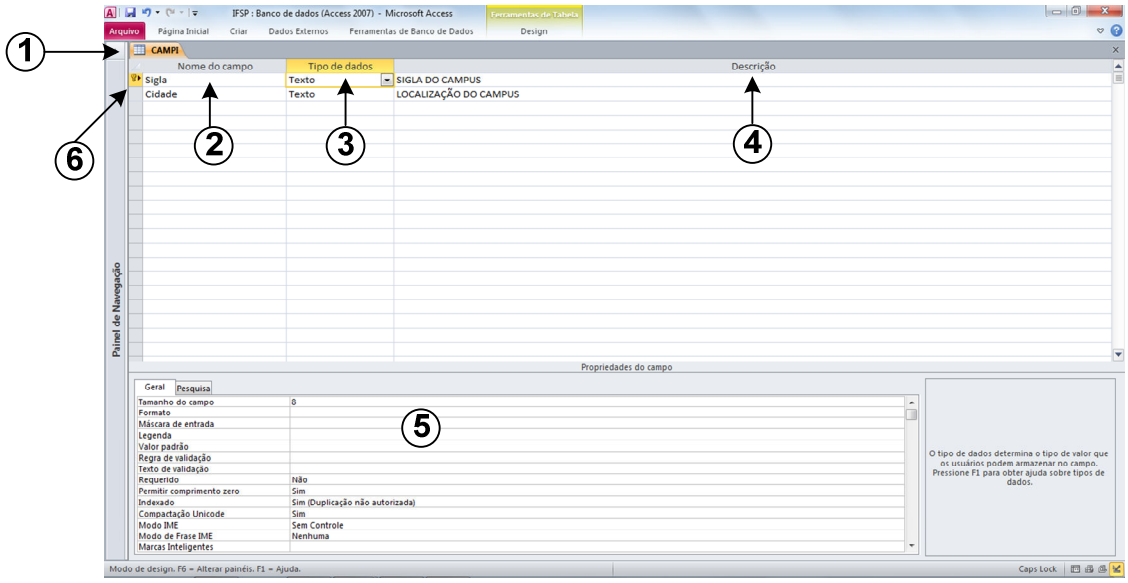


FIGURA 15 CRIAÇÃO DA TABELA *CAMPI* NO MODO DESIGN

- através de uma consulta de *Definição de Dados*, onde podemos escrever o comando SQL diretamente. A mesma tabela da figura 15 é agora criada com a instrução SQL (Figura 16).

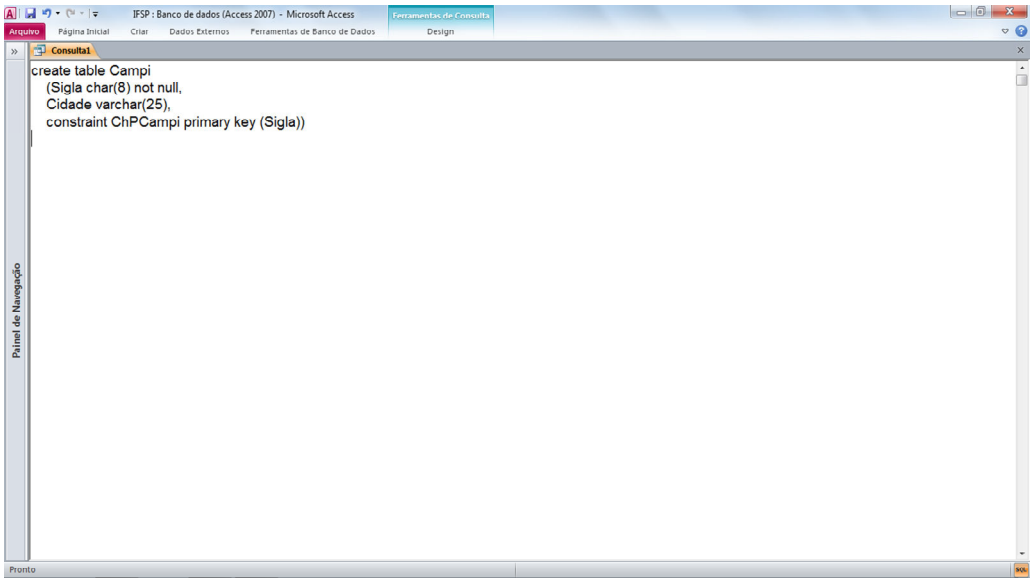


FIGURA 16 CRIAÇÃO DA TABELA *CAMPI* ATRAVÉS DE UMA CONSULTA *DEFINIÇÃO DE DADOS*

10.1.2. CONSULTAS

"Uma consulta, para o Access, é tão somente uma expressão SQL armazenada para uso posterior" CARVALHO (2006, p. 455).

Segundo SETZER e CORRÊA DA SILVA (2005, p. 229), o Access possibilita a criação de dois tipos de consulta: *seleção* e *ação*. O primeiro tipo permite recuperar

dados do banco de dados. O segundo tipo implementa comandos para a atualização – inclusão, exclusão e alteração – no banco.

Considerando esses tipos, encontram-se no Access as seguintes consultas:

- *Seleção*: tem por base a instrução SQL *select* e é usada para selecionar campos e registros das tabelas do banco de dados;
- *Acréscimo*: origina-se da instrução *insert into* e serve para adicionar registros às tabelas;
- *Exclusão*: construída a partir da instrução *delete*, é utilizada para excluir registros das tabelas do banco de dados;
- *Atualização*: é baseada na instrução *update* e permite alterar os valores dos campos em uma tabela;
- *Criar Tabela*: contém a instrução *create table* e possibilita a criação de uma tabela no banco de dados.

Todas as consultas anteriores podem ser realizadas através da interface abaixo (Figura 17), selecionando o do *tipo de consulta* desejada (item 2) ou diretamente através de instruções SQL, no *Modo SQL* (item 1).

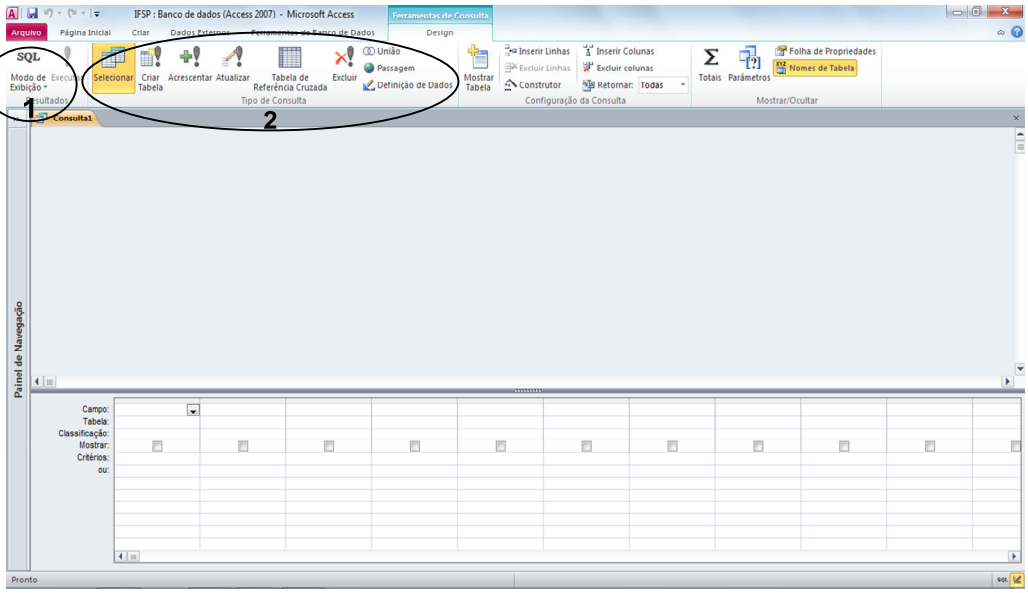


FIGURA 17 TIPOS DE CONSULTA(1) E MODO SQL (2)

Por exemplo, considerando a tabela criada na figura 16 (tabela *Campi*), suponha que seja necessário listar todos seus campos (colunas), porém em ordem ascendente pelo campo Cidade. Poderíamos montar essa consulta conforme ilustrado na figura 18 ou 19 – das duas maneiras obteríamos o mesmo resultado.

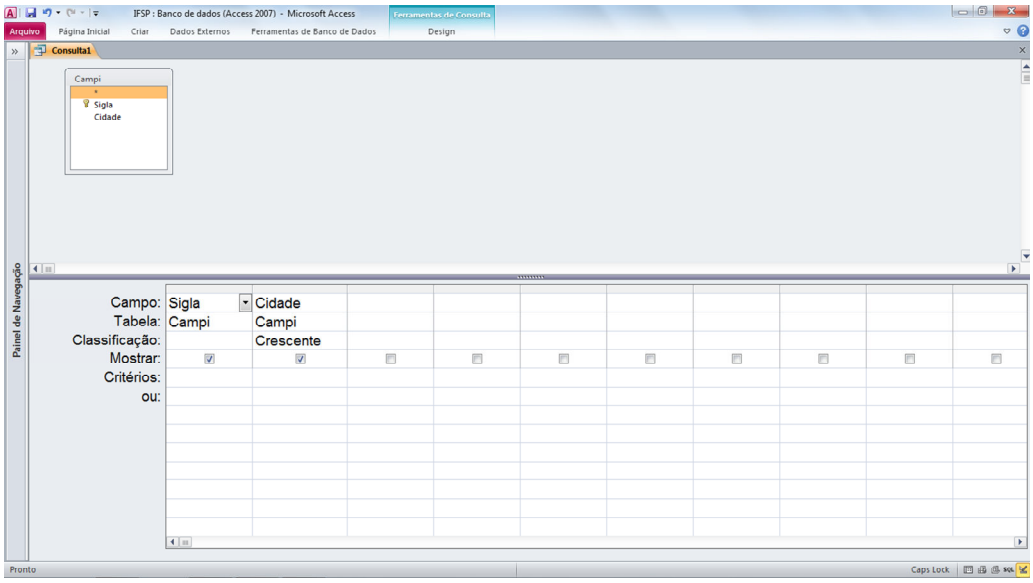


FIGURA 18 CONSULTA ATRAVÉS DO *MODO DESIGN*

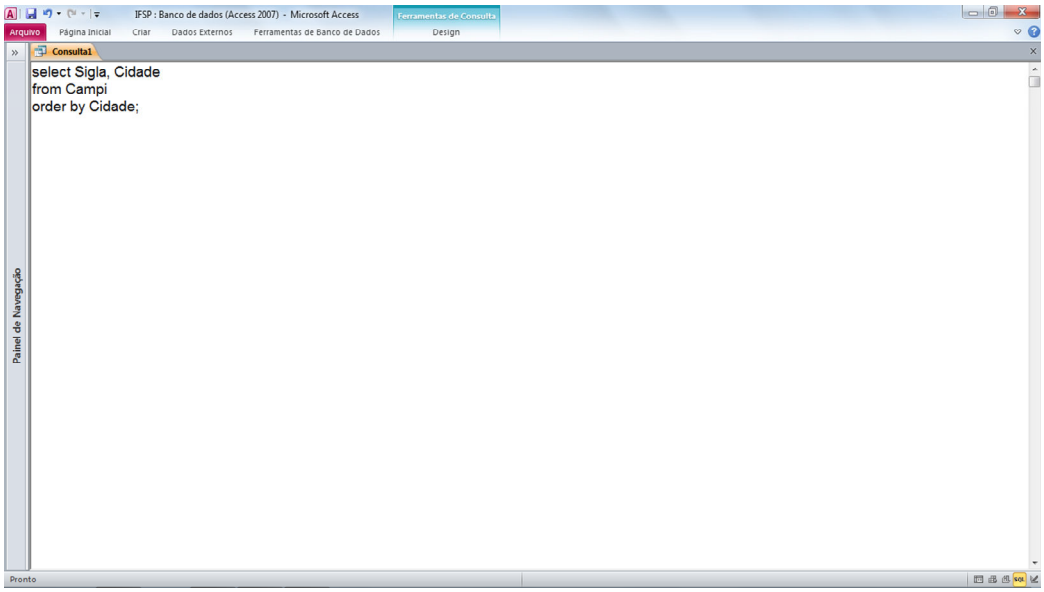


FIGURA 19 CONSULTA ATRAVÉS DO *MODO SQL*

10.1.3. FORMULÁRIOS

Um formulário é um objeto de banco de dados por meio do qual é possível exibir e inserir dados em tabelas. Em outras palavras, formulários servem de interface para inserção e visualização (em tela) de dados do banco de dados. Por exemplo, poderíamos montar um formulário (uma tela) para inclusão de novos *campi*, assim como para exibição daqueles já existentes na tabela.

10.1.4. RELATÓRIOS

Um relatório, que costuma estar vinculado a tabelas ou consultas, é uma representação impressa de dados. Ou seja, é usado para apresentar os dados (saída, geralmente impressa). Imagine, por exemplo, que haja a necessidade de imprimir uma lista com o nome de todos dos *campi*. Para atender a essa solicitação montaríamos um relatório que, além do nome de cada *campus*, poderia conter número da página, o logo da instituição e a data em que foi impresso.

10.1.5. MACROS

O Access permite a construção de comandos – denominados *macros* – para automatizar tarefas. Por exemplo, no formulário para a visualização e inserção de novos *campi,* poderíamos criar um botão de comando que, ao ser pressionado, automaticamente enviaria o relatório com o nome de todos os *campi* cadastrados para a impressora padrão.

10.1.6. MÓDULOS

O Visual Basic for Applications (VBA) possibilita o desenvolvimento de programas – ditos *módulos* – dentro do Access. Com o VBA podemos usar variáveis, estruturas de decisão (if, else, select case), estruturas de repetição (do while, do until, for next), functions, procedures e demais ferramentas disponibilizadas por uma linguagem de programação. Para ilustrar, embora seja um exemplo irreal, considere que devamos classificar os registros da tabela *campi* em “extenso” ou “curto” de acordo com quantidade de caracteres que compõem a string presente na coluna Cidade. Poderíamos criar um procedimento que percorresse a tabela *campi* e avaliasse cada

registro – primeiramente, efetuando a contagem de letras da cadeia de caracteres contida em Cidade e depois realizando a classificação.

10.2. INSTITUTO FEDERAL DE SÃO PAULO

O Instituto Federal de Educação, Ciência e Tecnologia (IFSP) é uma autarquia federal, vinculada ao Ministério da Educação (MEC), que possui autonomia administrativa e didático-pedagógica. Ele é uma instituição que oferta cursos regulares de educação básica (Ensino Médio), profissional (Cursos Técnicos) e superior (Bacharelados, Licenciaturas, Tecnologias, Pós-Graduação *lato-sensu* e *stricto-sensu*).

Atualmente, além da Reitoria, o Instituto Federal de São Paulo conta com vinte e oito unidades de ensino, sendo vinte e quatro *campi* convencionais e quatro *campi* avançados. Seu corpo docente e técnico-administrativo é constituído, em sua grande maioria, por servidores do quadro permanente de pessoal que é regido pelo Regime Jurídico Único (RJU).

Entre outros pontos, o RJU – que configura a Lei nº 8112/90 – estabelece que a forma de ingresso no órgão ocorrerá mediante realização de concurso público e que os novos servidores estarão sujeitos a avaliação desempenho em estágio probatório durante os primeiros trinta e seis meses de efetivo exercício, devendo ocorrer uma avaliação a cada doze meses.

10.3. IFSP E UTILIZAÇÃO DO ACCESS

De acordo com as normas internas do IFSP, especificamente a Resolução nº 93/2005, cabe à Diretoria de Recursos Humanos (setor pertencente à Reitoria) operacionalizar as avaliações dos servidores admitidos em todos os *campi* – desde a abertura do processo, quando da primeira avaliação, até a homologação do resultado, após a última avaliação.

Desta forma, é obrigatório que haja um controle mensal referente aos processos que devem ser abertos, os que estão em andamento, assim como aqueles que devem ser homologados.

Antes do Plano de Expansão da Rede Federal de Educação Tecnológica – plano que multiplicou o número de Institutos Federais em todo o território nacional – o Instituto Federal de São Paulo executava todo o processo de avaliações quase que de forma manual, contando apenas com os programas Microsoft Word (para montar relatórios) e Microsoft Excel (para controlar prazos). Entretanto, após o Plano de Expansão, essa situação tornou-se insustentável, visto que o número de contratações cresceu muito.

Então, para suprir essa necessidade de organizar dados, foi que se passou a utilizar o Microsoft Access.

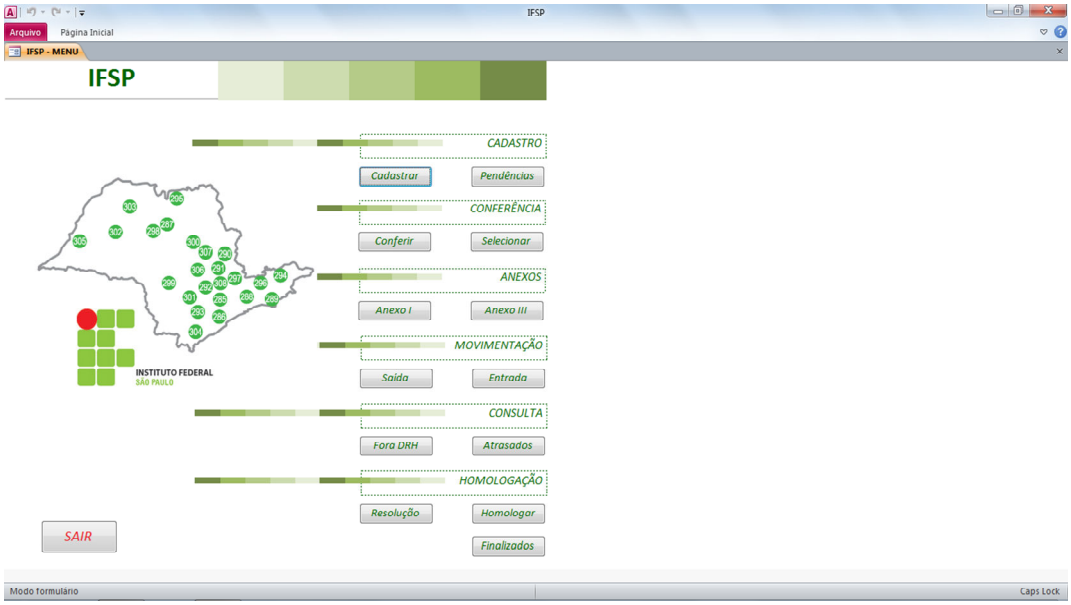
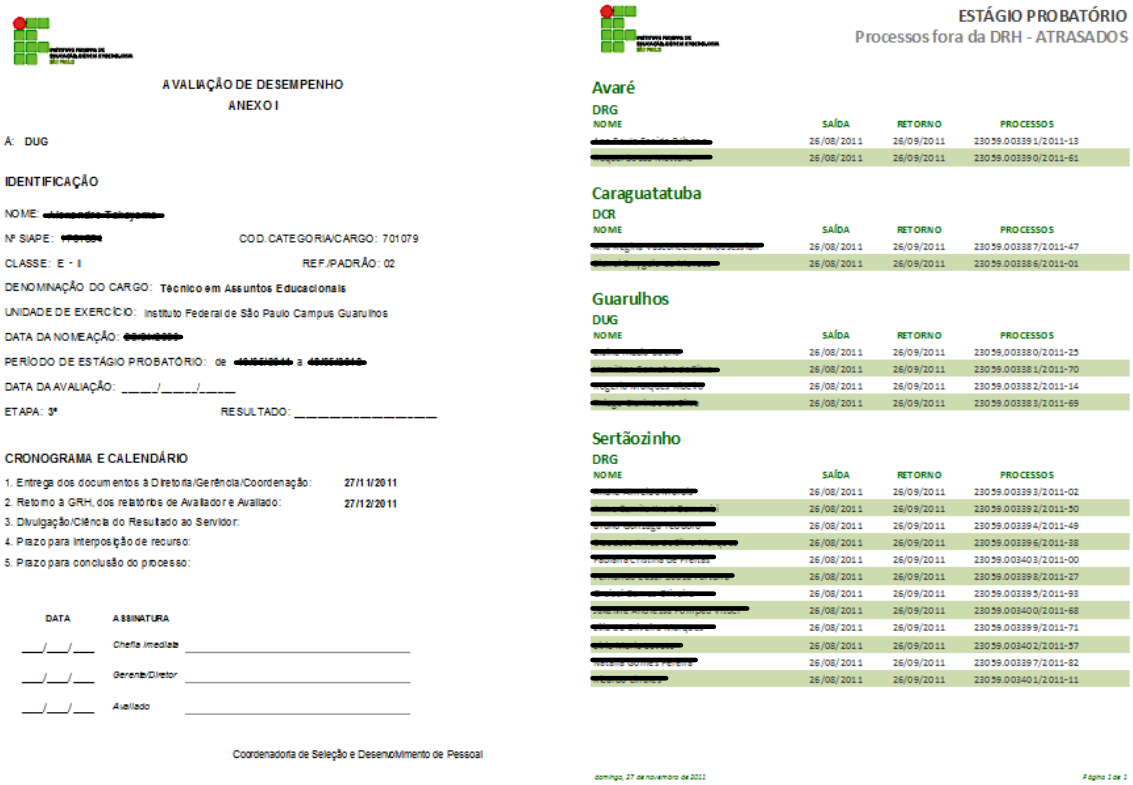


FIGURA 20 TELA INICIAL DO APLICATIVO *ESTÁGIO PROBATÓRIO*

Hoje são retirados do sistema todos os formulários necessários na montagem dos processos de avaliação, assim como é possível saber a situação de cada um deles. Consultas *ad-hoc* também podem ser executadas sempre que desejado.



AVALIAÇÃO DE DESEMPENHO
ANEXO I

À: DUG

IDENTIFICAÇÃO

NOME: [redacted]
Nº SIAPE: [redacted] COD.CATEGORIA/CARGO: 701079
CLASSE: E - I REF./PADRÃO: 02
DENOMINAÇÃO DO CARGO: Técnico em Assuntos Educacionais
UNIDADE DE EXERCÍCIO: Instituto Federal de São Paulo Campus Guarulhos
DATA DA NOMEAÇÃO: [redacted]
PERÍODO DE ESTÁGIO PROBATÓRIO: de [redacted] a [redacted]
DATA DA AVALIAÇÃO: _____/_____/_____
ETAPA: 3ª RESULTADO: _____

CRONOGRAMA E CALENDÁRIO

1. Entrega dos documentos à Diretoria/Gerência/Coordenação: 27/11/2011
2. Retorno à GRH, dos relatórios de Avaliador e Avaliado: 27/12/2011
3. Divulgação/Ciência do Resultado ao Servidor:
4. Prazo para interposição de recurso:
5. Prazo para conclusão do processo:

DATA	ASSINATURA	
__/__/__	Chefia Imediata	_____
__/__/__	Gerente/Diretor	_____
__/__/__	Avaliado	_____

Coordenadoria de Seleção e Desenvolvimento de Pessoal

ESTÁGIO PROBATÓRIO
Processos fora da DRH - ATRASADOS

Avaré
DRG

NOME	SAÍDA	RETORNO	PROCESSOS
[redacted]	26/08/2011	26/09/2011	23059.003391/2011-13
[redacted]	26/08/2011	26/09/2011	23059.003390/2011-61

Caraguatatuba
DCR

NOME	SAÍDA	RETORNO	PROCESSOS
[redacted]	26/08/2011	26/09/2011	23059.003387/2011-47
[redacted]	26/08/2011	26/09/2011	23059.003386/2011-01

Guarulhos
DUG

NOME	SAÍDA	RETORNO	PROCESSOS
[redacted]	26/08/2011	26/09/2011	23059.003380/2011-25
[redacted]	26/08/2011	26/09/2011	23059.003381/2011-70
[redacted]	26/08/2011	26/09/2011	23059.003382/2011-14
[redacted]	26/08/2011	26/09/2011	23059.003383/2011-69

Sertãozinho
DRG

NOME	SAÍDA	RETORNO	PROCESSOS
[redacted]	26/08/2011	26/09/2011	23059.003393/2011-02
[redacted]	26/08/2011	26/09/2011	23059.003392/2011-50
[redacted]	26/08/2011	26/09/2011	23059.003394/2011-49
[redacted]	26/08/2011	26/09/2011	23059.003396/2011-38
[redacted]	26/08/2011	26/09/2011	23059.003403/2011-00
[redacted]	26/08/2011	26/09/2011	23059.003398/2011-27
[redacted]	26/08/2011	26/09/2011	23059.003395/2011-93
[redacted]	26/08/2011	26/09/2011	23059.003400/2011-68
[redacted]	26/08/2011	26/09/2011	23059.003399/2011-71
[redacted]	26/08/2011	26/09/2011	23059.003402/2011-57
[redacted]	26/08/2011	26/09/2011	23059.003397/2011-82
[redacted]	26/08/2011	26/09/2011	23059.003401/2011-11

domingo, 27 de novembro de 2011 Página 1 de 1

FIGURA 21 EXEMPLOS DE RELATÓRIOS

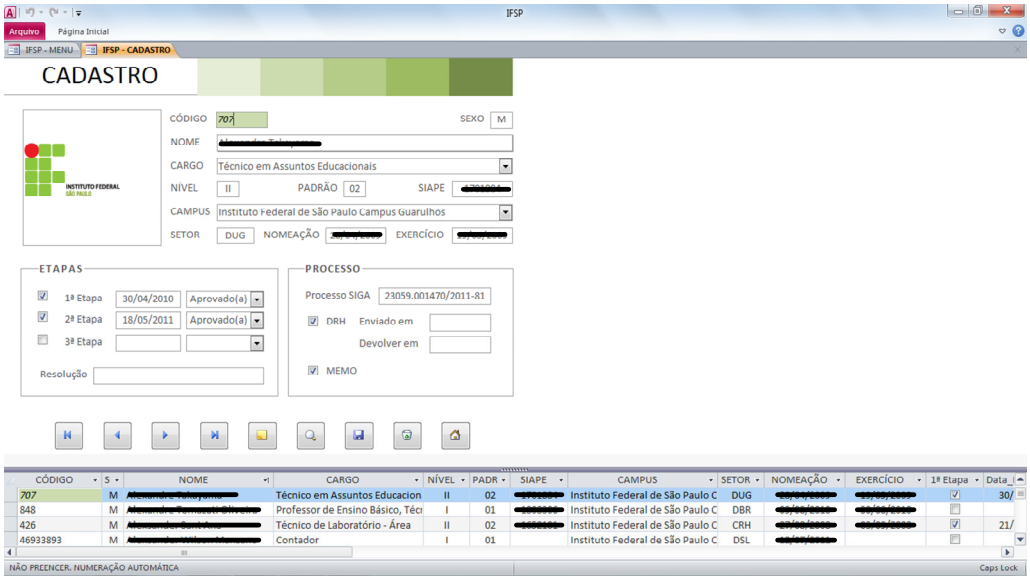


FIGURA 22 EXEMPLO DE FORMULÁRIO

11. CONSIDERAÇÕES FINAIS

A abordagem de bancos de dados apresenta vantagens se comparada com o processamento de arquivos tradicional, entre os benefícios estão a independência, a integridade e a consistência de dados, assim como o controle de redundância.

Um esquema de banco de dados diz respeito à estrutura do banco de dados. Esquemas podem ser descritos através de modelos de dados.

Da mesma forma que softwares de aplicação, um bancos de dados é desenvolvido por meio de projeto, sendo que esse compreende diversas etapas. Merecem destaque o projeto conceitual, no qual se emprega o Modelo Entidade-Relacinamento para descrever dados e sua semântica; e o projeto lógico, que é o mapeamento do modelo conceitual para o modelo de dados do SGBD que será adotado. Ainda hoje, a maior parte dos bancos de dados são relacionais, portanto, emprega-se nessa conversão o Modelo Relacional.

O Microsoft Access é um Gerenciador de Banco de Dados Relacional e como tal suporta a SQL – principal linguagem de consulta a bancos de dados relacionais. Embora ele componha a suíte de programas do pacote Microsoft Office, difere dos demais programas, como o Word ou o Excel, pois exige do usuário conhecimento da teoria de banco de dados. A vantagem de sua adoção está principalmente na facilidade de uso (devido à interface gráfica) e na disponibilização de recursos do VBA – linguagem de programação embutida no Access. É uma alternativa de solução para iniciar o tratamento de dados através da abordagem de banco de dados.

12. REFERÊNCIAS

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S.. **SISTEMA DE BANCO DE DADOS**. 3. ed. São Paulo: Makron Books, 1999.

ELMASRI, Ramez; NAVATHE, Shamkant B.. **SISTEMAS DE BANCO DE DADOS**. 6. ed. São Paulo: Addison Wesley, 2011.

DATE, C. J.. **INTRODUÇÃO A SISTEMAS DE BANCOS DE DADOS**. 8. ed. Rio de Janeiro: Elsevier, 2003.

SETZER, Valdemar W.; SILVA, Flávio Soares Corrêa da. **BANCOS DE DADOS**: APRENDA O QUE SÃO, MELHORE SEU CONHECIMENTO, CONSTRUA OS SEUS. São Paulo: Edgard Blücher, 2005.

OLIVEIRA, Marcos Raul de. **ADMINISTRAÇAO DE EMPRESAS COM ACCESS**. São Paulo: Digerati Books, 2005.

CARVALHO, Joao Antonio. **INFORMATICA PARA CONCURSOS**. 3. ed. Rio de Janeiro: Elsevier, 2006.

BRASIL. Instituto Federal de São Paulo. ESTATUTO. Disponível em: < http://www.ifsp.edu.br/index.php?option=com_content&view=article&id=76&Itemid=109>. Acesso em: 30 nov. 2011.

BRASIL. Instituto Federal de São Paulo. Resolução nº 93, de 15 de setembro de 2005. Dispõe sobre os procedimentos e instrumentos para Avaliação de Desempenho dos servidores Docentes e Técnicos Administrativos em Estágio Probatório. Disponível em: <http://www.ifsp.edu.br/index.php?option=com_phocadownload&view=category&id=82:resolues-2005&Itemid=149>. Acesso em: 30 nov. 2011.

BRASIL. Lei nº 8.112, de 11 de dezembro de 1990. Dispõe sobre o regime jurídico dos servidores públicos civis da União, das autarquias e das fundações públicas federais. Disponível em: < http://www.planalto.gov.br/ccivil_03/LEIS/L8112cons.htm >. Acesso em: 30 nov. 2011.