

PARTE I - INTRODUÇÃO A BANCO DE DADOS

CAPÍTULO I - CONCEITOS BÁSICOS

Introdução	01
1. Arquivo	02
2. Registro	02
3. Campo	03
4. Chave Primária	04
5. Chave Secundária.....	05
6. Chave Candidata.....	06

CAPÍTULO II - ORGANIZAÇÃO DE ARQUIVOS

1. Método De Acesso	07
2. Organização Seqüencial	09
3. Organização Serial.....	10
4. Organização Indexada	11

CAPÍTULO III - SGBD

1. Sistema Gerenciador de Banco de Dados - SGBD.....	13
2. Banco de Dados.....	13
3. Sistema em Banco de Dados.....	14

CAPÍTULO IV - OBJETIVOS DE BANCO DE DADOS

1. Independência de dados.....	15
2. Compartilhamento de dados.....	16
3. Menor redundância.....	16
4. Privacidade de dados	16
5. Segurança de dados	17
6. Tratamento de concorrência	17
7. Integridade de dados	18

CAPÍTULO V - LINGUAGENS DE BD

1. SQL.....	19
2. Autocontidas.....	22
3. Hospedeiras.....	22
4. Visuais.....	23
5. Linguagens para INTERNET.....	23
Conclusão.....	24

CAPÍTULO VI - BANCO DE DADOS RELACIONAL

1. Terminologia do Modelo Relacional.....	25
2. Regras de integridade.....	26
a. Integridade Declarativa.....	27
b. Integridade Procedural.....	29
c. Integridade Referencial.....	30
3. Operadores Relacionais.....	32
4. Propriedades Relacionais.....	33
5. Vantagens do Modelo Relacional.....	34

CAPÍTULO VII - ÁLGEBRA RELACIONAL

1. Estudo de caso	36
2. Generalidades	38
3. Operadores de Conjunto	39
a. União.....	40
b. Interseção.....	41
c. Diferença.....	42
d. Produto carteziano.....	43
4. Operadores Relacionais	44
e. Projeção.....	44
f. Seleção.....	45
g. Junção.....	46

PARTE II - PROJETO DE BANCO DE DADOS

Introdução	48
------------------	----

CAPÍTULO VIII - NÍVEIS DE ABSTRAÇÃO

1. Mundo Real	50
2. Modelo Descritivo	50
3. Modelo Conceitual	51
4. Modelo Operacional	51
5. Modelo Interno	52

CAPÍTULO IX - MODELO DE ENTIDADES E RELACIONAMENTOS (MER)

Generalidades	53
1. MER - Notação e terminologia.....	54
2. Regras para atribuição de nomes à entidades.....	55
3. Atributo	56
4. Dicionário de Dados.....	57
5. Relacionamento.....	59
6. Cardinalidade.....	60
7. Nome do relacionamento..	62
8. Papel	63
9. Sentenças	64
10. Integridade Referencial	65
11. Regras gerais para o MER	66
12. Vantagens e desvantagens do MER	67

CAPÍTULO X - TIPOS ESPECIAIS DE RELACIONAMENTO

1. Auto-relacionamento	68
2. Mais de um relacionamento	69
3. Relacionamento Múltiplo	70

CAPÍTULO XI - EXTENSÕES AO MER

1. Generalização	72
2. Especialização	73
3. Agregação	74
4. Variação do conceito de Agregação	75

CAPÍTULO XII - NORMALIZAÇÃO

1. Anomalias de Atualização	76
2. Terminologia	77
Dependência Funcional Completa (DFC)	78
Dependência Funcional Parcial (DFP)	79
Dependência Funcional Transitiva (DFT)	80
3. Notação para as estruturas de dados	81
4. Esquema de normalização	83
5. Relações não Normalizadas	84
6. Primeira forma normal (1FN)	85
7. Escolha da chave primária	86
8. Segunda forma normal (2FN)	88
9. Terceira forma normal (3FN)	89
Bibliografia.....	91

INTRODUÇÃO

No início da década de 60, foram lançados os primeiros sistemas gerenciadores de banco de dados (SGBD), tendo como principal proposta o aumento na produtividade nas atividades de desenvolvimento e manutenção de sistemas, até então realizadas de forma artesanal em linguagens de programação convencionais de primeira e segunda geração.

Oriundos do ambiente de mainframes, os SGBD tornaram-se mais populares e amigáveis com o advento da microinformática. Cada vez mais as fronteiras entre esses dois mundos estreitam-se e a concorrência pelo domínio do mercado de SGBD, tem levado seus diversos fabricantes a sofisticarem seus produtos. Cada nova versão lançada, incorpora novidades como interfaces gráficas, ferramentas de apoio ao desenvolvimento, utilitários para gerenciamento de BD e facilidades para extração de dados. Essa evolução vem tornando o trabalho de programadores, analistas e usuários menos artesanal, com reflexos na qualidade e produtividade.

A literatura classifica os SGBD como HIERÁRQUICO, REDE e RELACIONAL. Essa classificação representa a evolução desses produtos no curso da história. Atualmente, o mercado é dominado pelos SGBD RELACIONAIS e caminha para a colocação em escala comercial dos SGBD ORIENTADOS A OBJETOS.

Este texto introduz a teoria de BANCO DE DADOS, a partir de conceitos básicos da teoria de arquivos que perpetuaram-se na terminologia de banco de dados. Na sequência aborda superficialmente os modelos HIERÁRQUICO e REDE (por razões de mercado) e de forma mais aprofundada o MODELO RELACIONAL, o qual designaremos neste texto pela sigla SGBD-R.

CAPÍTULO I

CONCEITOS BÁSICOS

Para compreender com maior facilidade os conceitos relativos a BANCO DE DADOS é de suma importância revisar-mos alguns conceitos básicos referentes à teoria e terminologia de arquivos convencionais, haja vista, que os primeiros SGBD foram criados a partir do aperfeiçoamento de sistemas gerenciadores de arquivo, e ainda utilizam muito da base conceitual e da terminologia de arquivos.

1. ARQUIVO

Um arquivo é uma coleção de REGISTROS do mesmo tipo, ou seja, referentes a um mesmo assunto e com o mesmo formato padrão (layout). Constitui o componente do sistema no qual são armazenados os dados, que combinados através dos programas servem de base para a geração da informação desejada pelo usuário, através de relatórios e consultas on-line.

Um sistema de controle de notas, por exemplo, pode armazenar seus dados em diversos arquivos, cada um contendo informações sobre um determinado item do sistema: ALUNO, PROFESSOR, MATÉRIA, NOTA, etc.

Essas informações podem ser combinadas através de programas para gerar, por exemplo, o BOLETIM ESCOLAR, a PAUTA ou uma tela de CONSULTA DE NOTAS.

2. REGISTRO

Um registro é constituído por conjunto de campos valorados (contendo dados). Consiste na unidade de armazenamento e recuperação da informação em um arquivo. Geralmente, os registros de um arquivo possuem um formato padrão (layout), definido pela seqüência, tipo e tamanho dos campos que o compõem. Porém, algumas linguagens de programação permitem a criação de registros com layouts deferentes em um mesmo arquivo, recurso este que raramente é utilizado.

3. CAMPO

É a unidade básica formadora de um registro. Constitui a célula da informação. É a menor porção de um arquivo que pode ser referenciada por um programa.

Cada campo possui NOME, TIPO e TAMANHO. Os tipos de campo mais comuns são:

NUMBER	_ Armazena somente números _ Pode conter casas decimais _ Pode ser utilizado em operações matemáticas
CHAR ou ALFANUMÉRICO	_ Pode armazenar letras, números e caracteres especiais
DATE	_ Armazena datas fazendo consistência automática
MEMO ou LONG	_ Armazena textos em formato livre

A figura a seguir sintetiza os conceitos de ARQUIVO, REGISTRO e CAMPO:

ARQUIVO ALUNO

LAYOUT				
CAMPOS TIPO e TAM.	MATRICULA NUMBER (03)	NOME CHAR (30)	ENDEREÇO CHAR (50)	DT_NASC DATE
REGISTROS	001	José	SQS 308 ...	23/08/78
	002	Maria	QND 14	25/09/70
	003	Ana	SQN 410 ...	10/08/85

4. CHAVE PRIMÁRIA (PRIMARY KEY - PK)

A CHAVE PRIMÁRIA (ou simplesmente CHAVE) é o identificador único de um registo em um arquivo. Pode ser constituída de um campo (CHAVE SIMPLES) ou pela combinação de dois ou mais campos (CHAVE COMPOSTA), de tal maneira, que não existam dois registros no arquivo com o mesmo valor de chave primária.

Em regra, todo arquivo deve possuir uma chave primária, que permita a identificação inequívoca do registro, especialmente, para dar maior consistência aos processos de inclusão, alteração e exclusão de dados.

Para que não ocorram duplicatas nos valores da chave, os campos que a compõem são de PREENCHIMENTO OBRIGATÓRIO (NOT NULL).

Na escolha da chave primária de um arquivo deve-se buscar campos que possuam ESTABILIDADE no valor armazenado. A escolha do NÚMERO DO TELEFONE como chave de um cadastro de clientes, por exemplo, seria inadequada, por que esse valor pode mudar com frequência. Sem considerar que o cliente pode ter mais de um telefone...

Deve-se também evitar a escolha de campos que possam causar AMBIGÜIDADE em relação aos valores nele contidos. Nesse sentido, seria inadequado a escolha do campo NOME para chave de um cadastro de clientes, haja vista, que um mesmo nome pode ser escrito de várias formas. Por exemplo: LUÍS, LUIZ, LOUIS, LOYS, LUYS.

Se desejássemos cobrar uma fatura de um cliente com um nome como esse, a probabilidade de erramos o cliente seria grande. Além disso, a extensão do campo (30 ou mais caracteres) é um outro aspecto que aumenta a possibilidade de erros.

DICAS PARA ESCOLHA DA CHAVE PRIMÁRIA:

- _ Todo arquivo deve possuir uma chave primária.
- _ VALOR ÚNICO para cada registro.
- _ SIMPLES ou COMPOSTA.
- _ Campos de PREENCHIMENTO OBRIGATÓRIO.
- _ Valor ESTÁVEL.
- _ Não AMBÍGUO.
- _ PEQUENA EXTENSÃO (menor possível).
- _ De preferência CAMPOS NUMÉRICOS

5. CHAVE SECUNDÁRIA

A chave secundária pode ser formada por um campo ou pela combinação de campos (SIMPLES / COMPOSTA). É utilizada como parâmetro (filtro) para seleção de registros no arquivo em consultas, emissão de relatórios ou processos de atualização simultânea de um grupo de registros.

Por exemplo, para aumentarmos o valor do salário dos analistas em 10%, poderíamos utilizar o campo FUNÇÃO do arquivo CADASTRO DE FUNCIONÁRIOS como parâmetro (chave secundária) no processo de seleção dos registros a serem alterados.

Em síntese, a chave secundária é o campo ou combinação de campos do arquivo que permite a recuperação de mais de um registro no arquivo. Portanto, não possui a característica de unicidade proposta para a chave primária.

A figura a seguir ilustra os conceitos de CHAVE PRIMÁRIA e SECUNDÁRIA

ARQUIVO ALUNO

PK			
MATRICULA	NOME	ENDEREÇO	DT_NASC
001	José	SQS 308 ...	23/08/78
003	Maria	QND 14	25/09/70
002	Ana	SQN 410 ...	10/08/85
005	José	GAMA	05/04/76
.	.	.	.

Acesso via CHAVE SECUNDÁRIA (NOME) no arquivo ALUNO:

PROGRAMA X

INÍCIO....

.

.

SE NOME = "JOSÉ"
ENTÃO IMPRIMIR

.....

.

.

FIM

6. CHAVE CANDIDATA

Pode ocorrer uma situação em que mais de um campo satisfaça a condição de chave primária, constituindo duas ou mais CHAVES CANDIDATAS. Neste caso, o analista deverá eleger somente uma delas como CHAVE PRIMÁRIA, as demais permanecerão na condição de CANDIDATAS, indicando que tratam-se de campos de preenchimento obrigatório e com valores únicos para cada registro, o que será garantido através de mecanismos de integridade de coluna, que veremos no capítulo relativo a banco de dados.

A figura a seguir mostra um exemplo de arquivo com CHAVE CANDIDATA

ARQUIVO ALUNO

MATRICULA	NOME	ENDEREÇO	CPF
001	José	SQS 308 ...	72993246500
003	Maria	QND 14	12354789065
002	Ana	SQN 410 ...	09876587659
005	José	GAMA	28746503645
.	.	.	.
.	.	.	.

CHAVE PRIMÁRIA (indicated by an arrow pointing to the MATRICULA column)

CHAVE CANDIDATA (indicated by an arrow pointing to the CPF column)

CAPÍTULO II

ORGANIZAÇÃO DE ARQUIVOS

O tema organização de arquivos refere-se a forma como os registros são armazenados em um arquivo baseado em computador. Confunde-se com MÉTODO DE ACESSO, que consiste na forma como esses podem ser recuperados. A organização do arquivo determina os métodos de acesso que podem ser utilizados na recuperação dos registros, mas tratam-se de coisas distintas.

Apesar de este ser um assunto muito abrangente e com muitas variantes em termos de abordagem, trataremos de apenas três tipos de organização (SEQÜENCIAL, SERIAL E INDEXADA) e seus respectivos métodos de acesso. Essa escolha baseia-se na necessidade de discutirmos alguns conceitos essenciais para o estudo do modelo Relacional de banco de dados, que constitui o objeto principal desse texto.

1. MÉTODOS DE ACESSO

Para recuperarmos um registro em um arquivo, podemos utilizar acesso **SEQÜENCIAL** ou **DIRETO**.

O método **SEQÜENCIAL** de acesso é o mais tradicional e consiste em efetuar a leitura dos registros, um após o outro, comparando o ARGUMENTO DE PESQUISA, com o valor do campo CHAVE (primária ou secundária) no registro corrente, até encontrar os registros desejados ou o final do arquivo.

exemplo:

PROGRAMA Y

INÍCIO....

.

Repita até fim

ler registro

chave secundária (campo chave)

SE NOME = "JOSÉ"

ENTÃO IMPRIMIR

argumento de pesquisa

Fim repita (volte a ler)

.

FIM DO PROGRAMA

O método **DIRETO** consiste em recuperar o(s) registro(s) desejado(s), sem a necessidade de efetuar a leitura dos registros que o(s) antecede(m), o que pode ser feito através de um **ÍNDICE** (que abordaremos no item organização indexada) ou com o auxílio de um algoritmo de **RANDOMIZAÇÃO** que localiza o registro, calculando a posição ocupada pelo registro no disco, com base no valor do argumento de pesquisa, que deve ser um campo numérico.

Em ambos os casos, a localização do registro ocorre a cargo do gerenciador de arquivos, de maneira transparente para o programador, que só precisa escolher a organização adequada para o arquivo e fornecer no programa o argumento de pesquisa.

exemplo:

PROGRAMA Z

INÍCIO....

.

.

ABRIR ARQUIVO ALUNO INDEXADO POR NOME

.

NOME="JOSÉ" → argumento de pesquisa

LOCALIZAR REGISTRO → acesso direto (indexado)

SE ENCONTROU REGISTRO

ENTÃO IMPRIMIR

.

.

FIM DO PROGRAMA

2. ORGANIZAÇÃO SEQÜENCIAL

A ORGANIZAÇÃO SEQÜENCIAL caracteriza-se pela existência de uma CHAVE DE ORDENAÇÃO. Essa chave determina a ordem em que os registros são armazenados e pode ser SIMPLES ou COMPOSTA por dois ou mais campos. Geralmente, coincide com a chave primária, mas não obrigatoriamente.

A organização seqüencial somente permite o ACESSO SEQÜENCIAL.

A figura a seguir apresenta um arquivo com ORGANIZAÇÃO SEQÜENCIAL e CHAVE PRIMÁRIA(MATRICULA) distinta da CHAVE DE ORDENAÇÃO (NOME - ordem alfabética).

ARQUIVO ALUNO

chave primária **chave de ordenação**

↙ ↙

MATRICULA	NOME	ENDEREÇO	DT_NASC
001	Ana	SQS 308 ...	23/08/78
003	José	QND 14	25/09/70
002	José	SQN 410 ...	10/08/85
005	Maria	GAMA	05/04/76
.	.	.	.
.	.	.	.

3. ORGANIZAÇÃO SERIAL

Nesta forma de organização os registros são armazenados de acordo com a ordem de inclusão. o arquivo não possui chave de ordenação, portanto não existe preocupação com a ordem de armazenamento dos registros. No entanto, é sempre recomendável o arquivo possua uma chave primária.

A organização serial somente permite o ACESSO SEQÜENCIAL. Não deve ser utilizada em processos de exclusão e alteração de registros na modalidade batch (atualização em lote), pois degrada a performance.

É muito utilizada em processos de inclusão de registros onde não haja preocupação em manter a seqüência dos mesmos ("pools" de digitação). É também empregada no arquivo de dados que serve de base para a organização indexada, que estudaremos no próximo item.

A figura a seguir apresenta um arquivo com ORGANIZAÇÃO SERIAL. Note que ele não possui CHAVE DE ORDENAÇÃO.

ARQUIVO ALUNO

 **chave primária**

MATRICULA	NOME	ENDEREÇO	DT_NASC
005	Maria	SQS 308 ...	23/08/78
003	José	QND 14	25/09/70
002	Ana	SQN 410 ...	10/08/85
001	José	GAMA	05/04/76
.	.	.	.

4. ORGANIZAÇÃO INDEXADA

Nesta forma de organização, os registros são armazenados em um arquivo de dados com organização serial e para cada campo (ou combinação deles) através do qual se deseja obter acesso direto (indexado) deve-se criar um arquivo de índice (processo de indexação).

Um mesmo arquivo de dados pode possuir diversos arquivos de índice a ele associados. Porém, apesar da flexibilidade para a criação de índices, esse recurso deve ser utilizado com critério, pois a manutenção de muitos índices pode degradar a performance no processo de atualização do arquivo. Ou seja, ganha-se na consulta on-line, mas pode-se perder na atualização de dados.

O arquivo de índice é composto basicamente por duas colunas. A primeira corresponde ao campo utilizado no processo de indexação (endereço lógico) e a segunda armazena um valor (endereço físico) que serve como referência, para que o gerenciador de arquivos localize o registro no disco magnético.

Os registros dos arquivos índice são ordenados pelo endereço lógico. Portanto, se utilizarmos um algoritmo de leitura seqüencial em um arquivo indexado por nome, por exemplo, obteremos os registros em ordem alfabética, mesmo sendo o arquivo de dados um arquivo serial. Ou seja prevalece a ordem do índice. Porém nesse exemplo, a performance a performance do arquivo indexado seria menor, se comparada a de um arquivo seqüencial por nome.

Sempre que um arquivo índice for referenciado por um programa, ele será carregado para memória principal, o que torna desprezível o tempo de busca dos registros nesse arquivo. Além disso, o algoritmo utilizado na busca é o de pesquisa binária, o que reduz ainda mais o tempo.

Os índices constituídos com base no valor da chave primária ou candidata são conhecidos como ÍNDICES PRIMÁRIOS e os demais como ÍNDICES SECUNDÁRIOS.

Em resumo, a organização indexada é formada pela combinação de pelo menos um arquivo de dados e um ou mais arquivos de índice.

A figura a seguir apresenta o cenário da ORGANIZAÇÃO INDEXADA.

ARQUIVO ALUNO

ÍNDICE PRIMÁRIO

MATR	TSL
001	220
002	321
003	231
005	110
.	

ÍNDICE SECUNDÁRIO

NOME	TSL
Ana	321
José	220
José	231
Maria	110
.	331

TRILHA, SETOR E LADO DO DISCO

(endereço físico)

chave primária (endereço lógico)

TSL	MATR	NOME	ENDEREÇO	DT_NASC
110	005	Maria	SQS 308 ...	23/08/78
231	003	José	QND 14	25/09/70
321	002	Ana	SQN 410 ...	10/08/85
220	001	José	GAMA	05/04/76
331

CAPÍTULO III

SISTEMA GERENCIADOR DE BANCO DE DADOS - SGBD

A expressão BANCO DE DADOS, é coloquialmente empregada com os mais diversos significados, de tal sorte que, ao indagarmos de alguém sobre o BANCO DE DADOS com o qual trabalha em sua empresa, poderemos obter as seguintes respostas:

1. Trabalho com ORACLE, ACCESS, SQL SERVER, SYBASE, etc..
2. Trabalho com o banco de dados de PESSOAL, MATERIAL ou FINANÇAS;
3. Trabalho com o CADASTRO DE PESSOAL, SISTEMA DE VENDAS, etc.

Para evitar conflitos terminológicos, definimos a seguir três expressões, consagradas na literatura clássica, que seriam melhor aplicadas a cada uma das situações anteriores.

1. SISTEMA GERENCIADOR DE BANCO DE DADOS - SGBD

Essa expressão estará corretamente empregada, quando utilizada para designar o **SOFTWARE** utilizado para criar um BANCO DE DADOS. Portanto, tratando-se de SGBD estaremos nos referindo a produtos como ACCESS, ORACLE, SYBASE, SQL SERVER, ADABAS, etc.

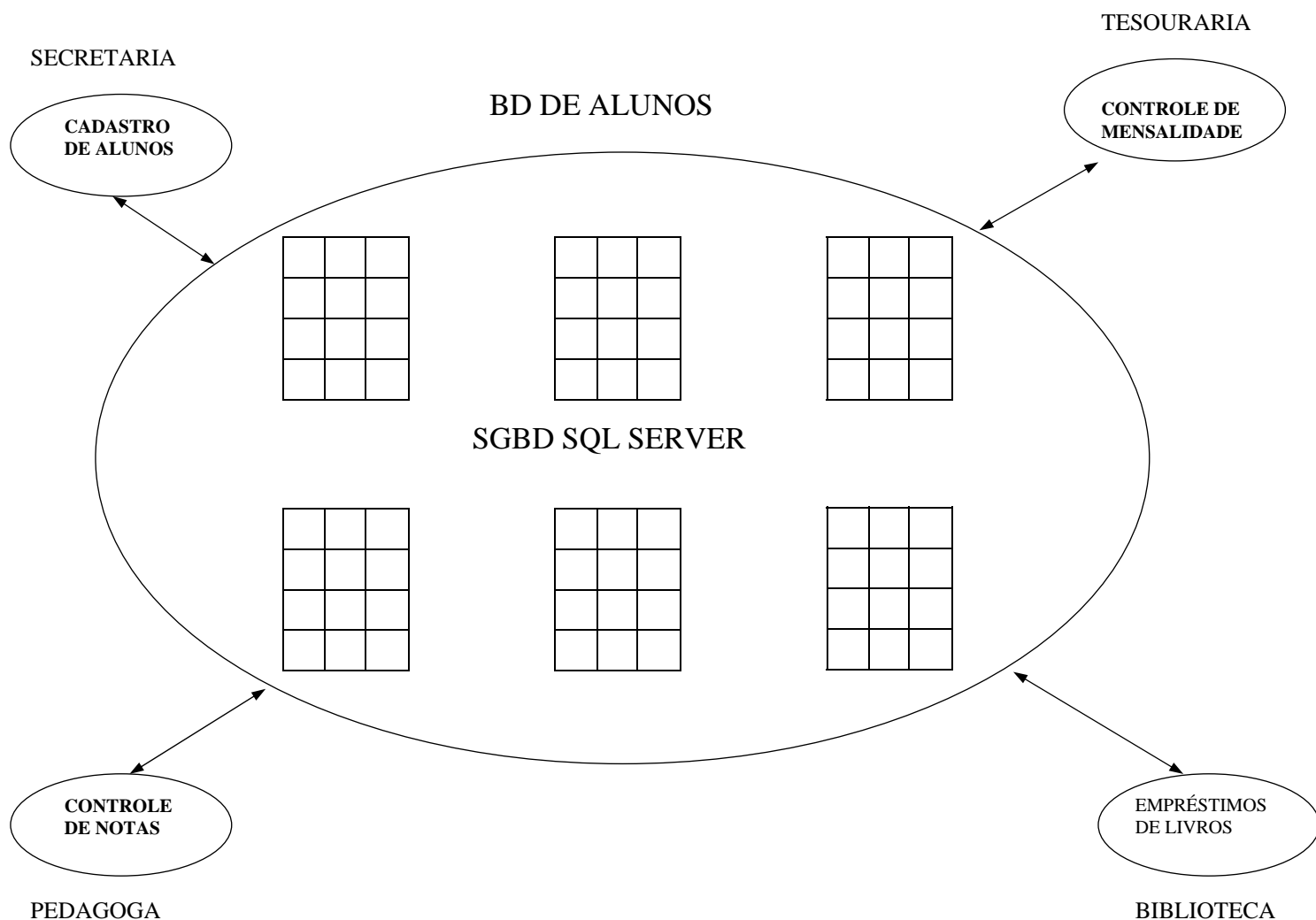
2. BANCO DE DADOS - BD

Esse enunciado refere-se a um conjunto de informações relacionadas, que são armazenadas no computador e recuperadas com a utilização dos recursos de um **SGBD**. Essas informações devem ser estruturadas, de tal maneira, que independam de aplicações específicas. Ou seja, um **BD** de **PESSOAL**, adequadamente estruturado, pode fornecer dados, tanto para um sistema de Folha de Pagamento, quanto para um sistema de Treinamento de Recursos Humanos.

3. SISTEMA EM BANCO DE DADOS - SBD

Essa expressão refere-se às **APLICAÇÕES** desenvolvidas para atender a necessidades específicas da empresa, que acessam um ou mais BD, para leitura ou atualização de informações. Tome como exemplo de aplicações específicas os sistemas de **folha de pagamento** e **Treinamento de Recursos Humanos**, citados no item anterior.

A figura abaixo ilustra um ambiente onde o BANCO DE DADOS de alunos foi estruturado para atender a quatro SISTEMAS distintos: CADASTRO DE ALUNOS, CONTROLE DE MENSALIDADES, EMPRÉSTIMO DE LIVROS e CONTROLE DE NOTAS. O BD foi montado utilizando os recursos do SGBD SQL SERVER.



CAPÍTULO IV

OBJETIVOS DE BANCO DE DADOS

O desenvolvimento da tecnologia de banco de dados tem se pautado por buscar alcançar, como objetivo permanente o aumento de produtividade nas atividades de desenvolvimento e manutenção de sistemas. Nesse sentido os fabricantes de SGBD vem dotando seus produtos com mecanismos que facilitam a adaptação dos BD às novas necessidades que surgem no dia a dia e que reduzem o trabalho de programação. Aliado a esses dois fatores existe toda uma filosofia que orienta os técnicos na escolha do melhor produto para a sua empresa e no trabalho de projeto de banco de dados.

Dessa filosofia destacamos, a seguir, alguns objetivos de BD, os quais um profissional deve ter em mente ao lidar com essa tecnologia.

1. INDEPENDÊNCIA DE DADOS

Os SGBD devem ser dotados de recursos que possibilitem a descrição das estruturas de dados (layout de arquivos e/ou tabelas) de forma independente dos procedimentos de manipulação (leitura e gravação) de dados no BD. Esse objetivo visa tornar transparente para os programas que acessam o BD as alterações que, por ventura, venham a ocorrer nas estruturas de dados, como por exemplo o acréscimo de um novo campo de informação ao banco. Da mesma forma, alterações em lógicas de programas que acessam o BD não devem afetar as estruturas de dados.

Quanto maior o grau de independência de dados, menor será o tempo em que o BD ficará fora de operação para atividades de manutenção como, por exemplo, recompilação.

Até hoje, a maneira mais eficiente adotada pelos fornecedores de SGBD para implementação desse objetivo foi a utilização do SQL (structured query language) nos produtos que seguem o Modelo Relacional. O SQL possui grupos de comandos específicos e independentes para as tarefas de criação e alteração de tabelas (DDL - data definition language) e leitura e atualização do BD (DML - data manipulation language).

2. COMPARTILHAMENTO DE DADOS

Consiste na reutilização dos dados do BD pelo maior número possível de aplicações dentro da empresa. Nesse sentido, os dados do BD devem ser muito bem planejados e estruturados. Portanto, este objetivo de banco de dados está mais ligado a atividade de análise e projeto de BD.

O compartilhamento de dados visa diminuir a **redundância de dados**, considerando-o como um recurso da empresa e não propriedade de setores isolados da organização.

Para implementar o compartilhamento de dados é necessário que a empresa disponha de **recursos de rede**, que permitam colocar o BD ao alcance dos diversos usuários. Além disso, é necessário que o SGBD possua um competente sistema de segurança, para que se estabeleça a **privacidade de dados**, através de mecanismos de restrição de acesso.

3. MENOR REDUNDÂNCIA DE DADOS

Redundância de dados consiste na repetição de um mesmo dado em diversos arquivos (tabelas) de um sistema, banco de dados, ambiente computacional ou empresa. Como exemplo, pode-se tomar a ocorrência do dado "NOME DO FUNCIONÁRIO", em bases de dados não compartilhadas dos sistemas de CADASTRO, FOLHA DE PAGAMENTO e TREINAMENTO de uma empresa.

A redundância é danosa para o ambiente computacional, pois aumenta os custos com o armazenamento de dados, com o pessoal para manutenção de sistema.

Além disso, a redundância gera inconsistência de dados, ou seja, o dado redundante extraído a partir de arquivos diferentes apresenta valores divergentes. Tal fato, pode afetar a credibilidade do usuário no sistema e no pessoal de informática.

4. PRIVACIDADE DE DADOS

O COMPARTILHAMENTO DE DADOS leva um grande número de usuários, com funções diversificadas na empresa, a acessar um mesmo banco de dados. Nesse contexto, o objetivo de **privacidade de dados** ressalta a preocupação que o projetista de BD deve ter em vedar o acesso de usuários não autorizados a informações sigilosas ou de acesso restrito.

Nesse sentido, o **sistema de segurança** dos SGBD, devem possuir meios para que o projetista possa definir perfis diferenciados de acesso ao BD, com a criação de grupos de usuários e atribuição de direitos de acesso a esses grupos, a partir da utilização de senhas.

5. SEGURANÇA DE DADOS

A segurança das informações armazenadas no BD pode ser encarada sob dois prismas: SEGURANÇA LÓGICA e SEGURANÇA FÍSICA.

A SEGURANÇA LÓGICA é alcançada com a utilização dos mecanismos de restrição de acesso disponíveis nos SGBD para implementação do objetivo de **privacidade de dados**, tais como senhas e sistemas de LOG e AUDIT que registram dados sobre as operações que são efetuadas no BD (data, hora, usuário, comando, etc.).

A SEGURANÇA FÍSICA dos dados é obtida a partir de utilitários e aplicativos que os fabricantes colocam em seus produtos, visando facilitar o trabalho de proteção aos dados contra danos físicos, que podem ser causados por falhas de hardware ou queda da rede. Nessa linha destacam-se as ROTINAS DE BACKUP, GRAVAÇÃO COM ESPELHAMENTO e SISTEMAS DE MONITORAÇÃO DE TRANSAÇÕES DISTRIBUÍDAS (TWO-PHASE-COMMIT).

6. TRATAMENTO DE CONCORRÊNCIA

Este objetivo de BD aborda o aspecto do acesso simultâneo de dois usuários a um mesmo conjunto de informações. O SGBD deve possuir mecanismos para a identificação e tratamento desses acessos concorrentes, para garantir a consistência das informações do BD no sentido de sua veracidade.

Os sistemas de bloqueio (LOCK) e desbloqueio (UNLOCK) são os mecanismos utilizados para evitar que uma informação que está sendo manipulada por um usuário ("USU1") seja alterada por outro ("usu2"). Enquanto o "USU1" dela se utiliza o 'USU2", não terá acesso a mesma ou o terá apenas para leitura e receberá um aviso do SGBD de que a informação está sendo acessada por outro usuário e pode ser modificada.

Existem vários níveis de LOCK. As opções variam conforme o produto (SGBD) analisado, sendo que os mais comuns ocorrem a nível de tabela, página (conjunto de registros) e linha (nível mais baixo).

Cabe lembrar que o nível de bloqueio influi na performance do SGBD em ambientes de missão crítica (altos índices de acesso concorrente), sendo que quanto menor o nível de LOCK, a performance tende a ser melhor. Ressalta-se que além desse, existem outros fatores que influenciam na performance do SGBD.

7. INTEGRIDADE DE DADOS

A integridade de dados refere-se a mecanismos que estão disponíveis nos SGBD, que garantem a consistência dos dados armazenados no SGBD, segundo parâmetros de validação, especificados no momento de criação do BD, em conjunto com as estruturas de dados.

Esse objetivo só se tornou disponível, como recurso do SGBD, com o advento dos modelos Relacionais e consta como pré-requisito para enquadramento de produtos nessa categoria de SGBD.

No capítulo dedicado aos SGBD relacionais trataremos esse assunto com maior riqueza de detalhes.

CAPÍTULO V

LINGUAGENS DE BANCO DE DADOS

As linguagens de banco de dados consistem na interface do usuário para interagir com o SGBD. Neste texto destacamos cinco modalidades de linguagens que são mais comumente utilizadas nessa interação: SQL, autocontida, hospedeira, visuais e aquelas voltadas para INTERNET. Esta é uma classificação meramente didática que objetiva apenas demonstrar formas diferenciadas de interação com o BD. Outros autores possuem diferentes classificações.

1. SQL (STRUCTURED QUERY LANGUAGE)

A linguagem SQL (anteriormente escrita SEQUEL) foi criada junto com o Sistema R, primeiro protótipo de SGBD-R, desenvolvido de 1974 a 1979 no IBM San Jose Research Laboratory. A versão original do SQL foi baseada em uma linguagem anterior chamada SQUARE. As duas linguagens são essencialmente a mesma, mas a SQUARE usa uma sintaxe bem mais matemática, enquanto a SQL é mais parecida com o inglês.

A linguagem SQL é mais do que somente uma linguagem de consulta, sem que isto se oponha ao “query” no seu nome. Ela fornece funções de recuperação e atualização de dados, além de criação, manutenção da estrutura de dados e controle do ambiente do BD.

É uma linguagem essencialmente interativa, porém pode ser embutida em outras linguagens procedurais (que neste texto chamamos linguagem hospedeira) para ser utilizada em programas batch ou on-line, que acessam o BD. Suas principais características são:

_ **Padrão ANSI** (American National Standard Institute). O ANSI estabeleceu-se como um “padrão de fato” de SQL para os fornecedores de produtos relacionais, que atualmente lideram o mercado. Este aspecto facilita a interoperabilidade entre BDs de diferentes fornecedores.

_ **Padrão de acesso.** Todo o acesso ao BD Relacional é feito em SQL, mesmo que embutida em outra linguagem.

_ **Interpretada** (não compilada), característica que provê maior grau de independência de dados aos BD relacionais, ou seja, faz com que a aplicação reconheça alterações nas estruturas de dados, sem necessidade de ser recompilada.

_ **DDL, DML e DCL.** O SQL possui esses três grupos de comandos, montados conforme a função do comando no banco de dados. Esta característica também relaciona-se à independência de dados, uma vez que pode-se descrever os dados (DCL) de forma independente das aplicações (DML).

_ **DDL (Data Definition Language).** Linguagem para definição de dados, que compreende os seguintes comandos SQL:

_ **CREATE** - Utilizado para criar objetos (tabela, índice, view, sequence, etc.) no BD.

Exemplo: Criação da tabela funcionário com os atributos matricula e nome.

```
CREATE TABLE funcionário  
    (matricula number(05)  
    nome char (30);
```

_ **ALTER** - Utilizado para alterar objetos do BD (adicionar colunas, modificar tipo de dados, adicionar integridade, etc.).

Exemplo: Adição da integridade de chave primária à tabela funcionário.

```
ALTER TABLE funcionário  
ADD CONSTRAINT PRIMARY KEY (matricula);
```

_ **DROP** - Exclui objetos do BD.

Exemplo: Exclusão da tabela funcionário do BD.

```
DROP TABLE funcionário
```

_ **DML (Data Manipulation Language)**. Linguagem para manipulação de de dados, que compreende os comandos para que o usuário interaja com os dados armazenados no BD.

SELECT - Comando de leitura, utilizado para que o usuário possa efetuar consultas (query) nas tabelas do banco de dados. É o comando mais poderoso do SQL, efetua as sete operações da álgebra relacional conforme veremos no capítulo sobre banco de dados Relacional. Seu formato básico é **SELECT-FROM-WHERE** (leia-de-onde). Pode ser combinado com os demais comandos SQL constituindo “sub-queries”. O resultado de todo comando SELECT é uma tabela, que pode conter uma, nenhuma ou N linhas.

Exemplo: Ler da tabela funcionário “matricula” e “nome”, onde o salário seja maior do que 500,00.

```
SELECT matricula, nome  
FROM funcionário  
WHERE salário > 500,00;
```

INSERT - Utilizado para incluir registros nas tabelas do banco de dados.

Exemplo: Inclusão de um registro na tabela funcionário.

```
INSERT INTO funcionário (matricula, nome)  
VALUES ( 20,'Maria do Carmo');
```

UPDATE - Utilizado para alterar dados nas tabelas do banco de dados.
Exemplo: Alteração no salário do funcionário de matrícula igual a 20.

```
UPDATE funcionário  
SET salário=1200  
WHERE matricula=20;
```

DELETE - Utilizado para excluir registros das tabelas do banco de dados.

Exemplo: Exclusão do funcionário de matrícula igual a 20.

```
DELETE funcionário  
WHERE matricula=20;
```

2. LINGUAGEM AUTOCONTIDA

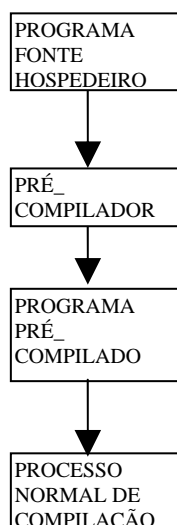
Esta modalidade de linguagem é a extensão procedural do SQL, que nos SGBD-R é utilizada para desenvolvimento de programas que ficam residentes no banco de dados (TRIGGERS, STORED PROCEDURES, FUNCÇÕES). Acrescenta ao SQL interativo estruturas de decisão (IF-THEN-ELSE) e repetição (LOOP, FOR e/ou DO WHILE). É uma linguagem proprietária (Cada SGBD possui a sua). Os programas escritos nessa linguagem geralmente assemelham-se a programas PASCAL.

3. LINGUAGEM HOSPEDEIRA

São linguagens procedurais de 3ª geração (notadamente o COBOL) utilizadas como hospedeiras (host) de comandos próprios de banco de dados. Linguagens hospedeiras foram muito utilizadas nos SGBD dos modelos Hierárquicos e Rede, dado que nestas gerações de SGBD ainda não existia o SQL com toda a sua simplicidade e potencialidade. Por outro lado, imperava uma forte cultura nas linguagens COBOL, PL/1, FORTRAN, etc.... que foi aproveitada pelos fabricantes de SGBD, facilitando a introdução dessa nova cultura.

Os SGBD que utilizam linguagens hospedeiras, possuem um software PRÉ-COMPILADOR, que é inserido na rotina de compilação do fonte do programa hospedeiro, para converter os comandos de BD (estranhos à linguagem HOST) em linguagem objeto ou chamadas (CALL) de rotinas intelegíveis pelo compilador a linguagem.

Esquema de compilação com linguagem hospedeira:

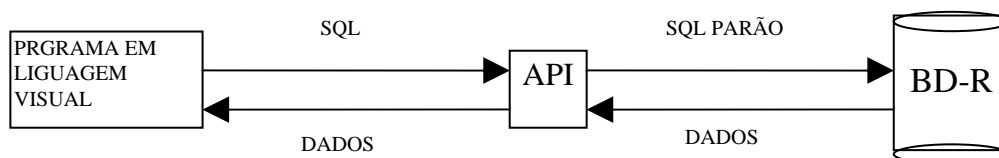


4. LINGUAGEM VISUAL

As linguagens visuais atualmente dominam o ambiente de desenvolvimento para a arquitetura Cliente/Servidor. Nessa arquitetura, são utilizadas para desenvolver a interface Cliente da aplicação. Recebem a denominação de FRONT-END. Geram aplicações para ambiente gráfico, padrão WINDOWS. São orientadas a eventos e em sua maioria, baseiam-se na tecnologia de Orientação a Objetos, apresentando recursos como classe, objeto, herança, polimorfismo, etc.. Utilizam o SQL como linguagem para acesso aos bancos de dados relacionais, através de APIs (Application Program Interface) nativas ou genéricas (ex: ODBC e ODAPI). Como exemplos de linguagem dessa categoria podemos citar: DELPHI, VISUAL BASIC, POWER BUILDER, CENTURA, etc..

Nossa maior preocupação neste texto é chamar a atenção do leitor para o fato de que essas linguagens são FRONT_ENDs de SGBD relacionais. Apesar de serem orientadas a objeto, não devem ser confundidas com os BD relacionais ou Orientados a Objeto, este último, ainda é uma tecnologia que ocupa um espaço restrito no mercado.

Esquema de acesso a BD relacional com linguagem visual:



5. LINGUAGENS PARA INTERNET

Nesse grupo de linguagens, incluem-se aquelas utilizadas para manipular, formatar, manipular e apresentar as informações que são acessadas na INTERNET através dos Browsers, das quais, destacam-se: HTML, JAVA SCRIPT, ASP e JAVA.

CONCLUSÃO

O banco de dados constitui a parte estática da aplicação. A dinâmica de um sistema é dada pelas linguagens que armazenam, recuperam e manipulam as informações contidas no BD e existem diversas maneiras para se executar essas tarefas.

A evolução das linguagens aponta para utilização de produtos não proprietários e que privilegiem a reutilização do código, além dos dados que já são amplamente reutilizados através da tecnologia relacional. Isso porque, o código contém a inteligência das aplicações (regras do negócio) que, em última análise, refletem o conhecimento da empresa a respeito do seu negócio. A reutilização do código pode trazer, como benefícios imediatos, a redução no tempo e custo de desenvolvimento e manutenção de novas aplicações, a padronização de processos e a melhoria constante na qualidades dos sistemas gerados.

A medida que surgem novas tecnologias para acesso às informações do BD, muitos sistemas antigos permanecem como legados, porque são sistemas críticos, de difícil substituição, ou porque são satisfatórios e, em análise de custo / benefício, tem o seu redesenvolvimento desaconselhado. Além disso, as necessidades de acesso às informações estão se diversificando, as empresas estão abrindo suas fronteiras e oferecendo acesso direto do cliente a suas bases de dados. Como resultado, é comum observarmos ambientes de sistemas heterogêneos, desenvolvidos com tecnologias diferentes, em termos de linguagens e banco de dados, onde observamos COBOL convivendo com DELPHI, VB, HTM, JAVA, etc..

Nesse contexto, também surge como uma tendência o desenvolvimento de aplicações em três ou “N” camadas. O foco principal dessa tecnologia é maximizar a reutilização de código, através do desenvolvimento das regras do negócio em componentes reutilizáveis, que obedeçam a padrões de mercado (CORBA, COM/DCOM, EJB) e que funcionem como linguagens universais, podendo ser acionados a partir de interfaces clientes desenvolvidas em diferentes linguagens.

Em síntese, o bancos de dados relacionais tornaram-se grandes repositórios de dados, constituindo uma tecnologia madura que tem atendido satisfatoriamente o mercado. Do lado das aplicações, observa-se uma tendência a diversificação de tecnologias, sem descuidar dos benefícios que podem trazer a reutilização das regras de negócio.

CAPÍTULO VI

BANCO DE DADOS RELACIONAL

O Modelo Relacional de Banco de Dados, utiliza a teoria de conjuntos como base conceitual para a formulação de seus conceitos. Esse pressuposto facilita o entendimento por parte do usuário e possibilita a representação do mundo real de forma mais natural.

O Modelo Relacional, começou a ser divulgado a partir de 1970, por E. F. Codd, um cientista da IBM, que utilizou o SISTEMA-R como produto experimental para a comprovação da teoria Relacional, publicada em uma série de artigos, que apresentaram os requisitos desse modelo em doze regras atualmente seguidas pelos Sistemas Gerenciadores de Banco de Dados Relacionais (SGBD-R).

As doze regras de Codd, foram reeditadas por diversos autores que escreveram sobre o modelo Relacional. Em nossa pesquisa bibliográfica para elaboração desse material, notamos que, existem interpretações ambíguas e até contraditórias em relação a essas regras. Portanto, para notar o estudo do modelo Relacional, adotamos a abordagem de C. J. DATE, que apresenta o Modelo Relacional como possuindo as seguintes características fundamentais, que o distingue dos demais modelos:

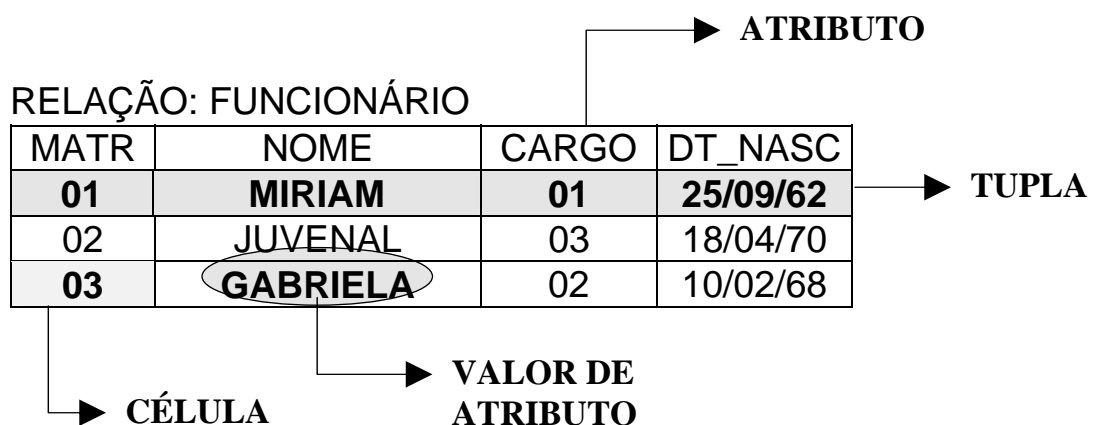
- _ Estrutura de dados tabular
- _ Regras de integridade
- _ Operadores relacionais
- _ Utilização do SQL (Structured Query Language)

O modelo Relacional, assim como seus antecessores, nasceu no ambiente dos computadores de grande porte (mainframe). Sofreu restrições ao uso, por demandar muita memória principal para alcançar uma performance (tempos de resposta) que o tornasse comercialmente viável. Ganhou força a partir do início a década de 80, com a revolução tecnológica provocada pela produção em larga escala dos microcomputadores PC, o que propiciou o barateamento do hardware.

Atualmente o modelo relacional é um padrão seguido, praticamente por todos os fornecedores de SGBD do mercado, Dentre os quais destacam-se: ORACLE, SYBASE, MYCROSOFT (SQL SERVER e ACCESS), INFORMIX e IBM DB/2.

1. TERMINOLIGIA DO MODELO RELACIONAL

- a. Os SGBD RELACIONAIS representam os dados sob a forma de TABELAS bidimensionais (linhas X colunas), denominadas **RELAÇÕES**.
- b. As linhas das tabelas são conhecidas como **TUPLAS** e as colunas como **ATRIBUTOS**.
- c. O número de atributos (colunas) de uma relação (tabela) determina o **GRAU DA RELAÇÃO**. Portanto uma relação com quatro colunas possui grau quatro.
- d. A interseção linha X coluna de uma tabela denomina-se **CÉLULA**.
- e. O conteúdo de uma célula denomina-se valor de atributo .
- f. Cada célula de uma tabela relacional comporta apenas um valor de atributo, característica a qual designa-se por **ATOMICIDADE** (valor atômico).
- g. O conjunto de valores possíveis para um atributo de tabela denomina-se **DOMÍNIO**. Por exemplo, o domínio para o atributo cargo pode ser definido como: Valor numérico entre 1 e 10.



2. REGRAS DE INTEGRIDADE

Integridade de dados é o conjunto de parâmetros (regras do negócio) previamente estabelecidos e criados no banco de dados, aos quais os dados são submetidos, para garantir que de um processo de atualização não resultem dados inconsistentes.

Uma das características mais fortes dos SGBD-R, está em oferecer mecanismos para a criação de regras de integridade diretamente no banco de dados. Nesse ponto a grande vantagem em relação aos demais modelos (Hierárquico e Rede), consiste no gerenciamento automático e centralizado de rotinas de integridade pelo SGBD, do que decorrem fatores como a eliminação de códigos redundantes e maior segurança no que se refere à consistência das informações.

Por outro lado, a possibilidade de definir integridade no BD, não descarta a hipótese de mantê-la no fonte da aplicação que acessa o BD. Na arquitetura Cliente/Servidor, essa prática é muito corriqueira e pode trazer significativos ganhos de performance.

As regras de integridade de dados podem ser implementadas nos SGBD-R de forma **DECLARATIVA** ou **PROCEDURAL**:

a. INTEGRIDADE DECLARATIVA

A integridade declarativa é implementada no BD, através de parâmetros opcionais da linguagem de definição de dados (DDL). Os tipos mais comuns de integridade declarativa são: **CHAVE PRIMÁRIA**, **DOMÍNIO** e **INTEGRIDADE REFERENCIAL**.

A integridade de **CHAVE PRIMÁRIA** garante que a chave primária da tabela não **contenha** valores em duplicata e nem valor NULO.

A integridade de **DOMÍNIO** permite restringir o universo de valores válidos para uma coluna.

A integridade **REFERENCIAL** garante o sincronismo de valores entre a chave estrangeira (foreign key) e a respectiva chave primária. Esse tipo de integridade será tratado com maiores detalhes no item “c” deste capítulo.

Na DDL do ORACLE, por exemplo, o comando CREATE apresenta as seguintes opções de integridade declarativa:

- _ PRIMARY KEY - Garante a integridade de chave primária.
- _ NOT NULL - Torna o campo de preenchimento obrigatório.
- _ CHECK - Permite a integridade de domínio.
- _ UNIQUE - Evita a ocorrência de valores em duplicata.
- _ FOREIGN KEY - Implementa a integridade referencial.

Exemplo:

```
CREATE TABLE funcionário
      (matricula number(05) PRIMARY KEY
       nome char (30) NOT NULL
       sexo char (01) CHECK sexo = 'F' or 'M';
```

No exemplo, o ORACLE encarrega-se da integridade de chave primária (PRIMARY KEY), da condição de campo obrigatório (NOT NULL) e da integridade de domínio (CHECK), todas especificadas de forma declarativa. Nenhuma linha de código é necessária nos programas que acessam BD para garantir essas integridades.

b. INTEGRIDADE PROCEDURAL

A Integridade Procedural apresenta-se sob a forma de um programa, cuja lógica é escrita pelo programador, na linguagem procedural nativa do SGBD. Esse tipo de integridade supre as necessidades não cobertas pelos parâmetros de integridade declarativa.

No ORACLE a integridade procedural pode ser criada através de TRIGGERS, STORED PROCEDURES ou FUNÇÕES DO USUÁRIO. Estes elementos são escritos em PL/SQL que é a extensão procedural do SQL desse SGBD.

Um TRIGGER (gatilho) é criado para disparar, automaticamente, sempre que o SGBD detectar a ocorrência de um ou mais comandos de acesso a tabela.

Exemplo:

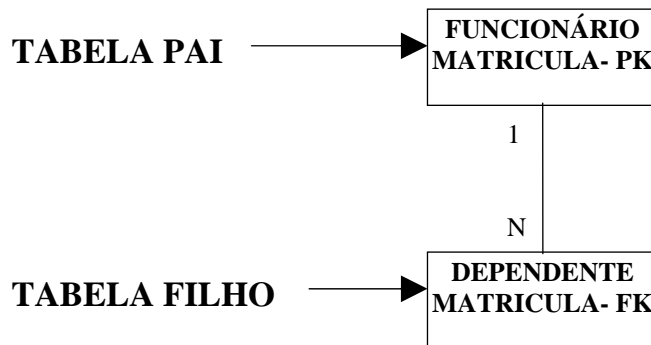
```
CREATE TRIGGER atualiza_saldo
AFTER INSERT ON TABLE lançamentos
BEGIN
    UPDATE Tab_saldo
    SET saldo_atual=saldo_atual + valor_lançamento;
END;
```

No exemplo, sempre que um registro for incluído na tabela “lançamentos” o trigger dispara e atualiza o “saldo_atual” na tabela “tab_saldo”.

Nem todos os SGBD possuem integridade procedural. Esse recurso é mais frequente nos SGBD de maior porte como ORACLE, DB/2, INFORMIX, SQL SERVER, etc..

c. INTEGRIDADE REFERENCIAL

A Integridade Referencial é o mecanismo dos SGBD-R que, no processo de atualização do BD, mantém o sincronismo entre duas tabelas relacionadas, em relação aos valores da chave estrangeira e da respectiva chave primária.



A integridade referencial evita a ocorrência de registros orfãos no banco de dados, ou seja, registros “filhos” sem a correspondente linha de referência na tabela “pai”.

Os SGBD_R que seguem o padrão SQL ANSI/92, suportam a integridade referencial de forma declarativa. Possuem ainda ações referenciais, que propagam atualizações e exclusões efetuadas na tabela pai para a tabela filho.

As ações referenciais propiciam, por exemplo, que a exclusão de um registro pai provoque a exclusão automática de seus respectivos filhos (exclusão em cascata), ou que a alteração no valor de uma chave primária reflitam automaticamente para os registros que a referenciam (atualização em cascata).

Exemplo:

```
CREATE TABLE funcionário
(matricula number(05) PRIMARY KEY
 nome      char   (30)
 sexo     char   (01));

CREATE TABLE dependente
(id_dependente      number(05) PRIMARY KEY
 nome_dependente   char   (30)
 data_nascimento   date
 matricula_funcionário number(05) FOREIGN KEY
                    REFERENCES funcionário.matricula
                    ON DELETE CASCADE);
```

O exemplo apresenta a integridade referencial, declarada na tabela “dependente”, indicando que o campo “matricula_funcionário” dessa tabela, refere-se ao campo “matricula” da tabela “funcionário”. Com essa declaração o SGBD garante que a inclusão de um “**DEPENDENTE**”, somente será válida caso exista o “**FUNCIONÁRIO**” correspondente.

Por outro lado, a cláusula “**ON DELETE CASCADE**” indica que sempre que for excluído um registro da tabela “funcionário”, o SGBD deve excluir automaticamente os registros da tabela dependente a ele relacionados.

3. OPERADORES RELACIONAIS

Os Operadores Relacionais constituem mecanismos do SGBD-R para recuperação de informações no Banco de Dados. Inserem-se no contexto da Álgebra Relacional que possui sete operadores, sendo três relacionais (PROJEÇÃO, SELEÇÃO e JUNÇÃO) e quatro operadores tradicionais de conjunto (UNIÃO, INTERSEÇÃO, DIFERENÇA e PRODUTO CARTEZIANO).

Para que um SGBD seja considerado relacional basta que possua apenas os operadores relacionais. Os operadores de conjunto podem ser simulados a partir dos primeiros.

No SQL/ANSI, os sete operadores são implementados por variações nas cláusulas do comando SELECT.

No próximo capítulo trataremos dos operadores relacionais com exemplos da aplicação de cada um deles.

4. PROPRIEDADES RELACIONAIS

As Propriedades relacionais são considerações óbvias, porém elucidativas a respeito do funcionamento e da filosofia que norteia o desenvolvimento dos SGBD-R. Essas propriedades derivam da teoria de conjuntos e algumas se sobrepõem ou confirmam as regras de integridade.

a. Uma tabela não deve possuir duas linhas iguais. Isto se explica pelo fato de que as linhas são componentes de um conjunto (a tabela) e se faz necessário poder distinguir os elementos de um conjunto. Assim sendo, pelo menos um atributo componente da linha deve possuir um valor que a diferencie das demais. Nos modelos relacionais o diferencial mínimo entre duas linhas de uma tabela é a chave primária.

b. Toda a tabela de um BD relacional deve possuir chave primária. Essa propriedade decorre da anterior. Atualmente, todos os SGBD-R disponíveis no mercado mantêm automaticamente a unicidade da chave primária. Por outro lado, alguns produtos relacionais permitem a criação de tabelas sem PK, deixando a critério do analista a sua declaração ou não, o que contraria esta propriedade mas atribui maior flexibilidade ao produto.

c. Cada tabela deve possuir um nome próprio, distinto das demais tabelas do mesmo banco de dados. Essa propriedade também deriva da teoria de conjuntos, já que as tabelas são componentes do conjunto BD. Ressalta-se que em banco de dados distintos duas tabelas podem ter o mesmo nome.

d. Cada atributo de uma mesma tabela deve possuir um nome diferente. Por outro lado, o mesmo atributo pode aparecer em outra tabela com o mesmo nome ou com nome diferente (sinônimo).

e. Os SGBD-R somente operam com estruturas de dados de formato tabular, normalizadas pelo menos em 1FN (1ª forma normal), onde a principal característica é a atômidade, ou seja, ocorrência de apenas um valor de atributo para cada célula da tabela. Esse nível de normalização é exigido para tornar possível a aplicação da Álgebra Relacional para recuperar informações contidas nas tabelas do BD. Níveis mais altos de normalização (2FN a 3FN) são úteis para diminuir a redundância, melhorar a consistência e integridade dos dados.

f. A ordem das linhas e colunas na tabela é irrelevante, pois pode ser facilmente modificada nas consultas, através dos recursos da linguagem SQL (Structured Query Language).

g. Os SGBD-R devem ser capazes de tratar, de maneira diferenciada o valor NULO (NULL), que indica ausência de valor para um atributo em determinada linha. Nulo corresponde na teoria de conjuntos a conjunto vazio e é diferente de zero ou branco.

5. VANTAGENS DO MODELO RELACIONAL

As vantagens em relação aos sistemas de arquivos convencionais e SGBD Hierárquicos e Rede são:

- a. Linguagem SQL interativa e muito próxima da linguagem natural escrita (inglês);
- b. Facilidade no entendimento da estrutura de dados tabular;
- c. Maior possibilidade de utilização direta pelo usuário final;
- d. Centralização da integridade no BD.
- e. Redução no tamanho dos códigos de programa;
- f. Maior integridade e consistência de dados;
- g. Maior segurança;
- h. Maior flexibilidade para acréscimo de novas informações no BD;
- i. Possibilidade de criar gatilhos (TRIGGERS) e procedimentos armazenados (STORED PROCEDURE);
- j. Maior Produtividade.
- l. Padronização dos produtos facilitando a difusão e preservação da cultura relacional.

CAPÍTULO VII

ÁLGEBRA RELACIONAL

A Álgebra Relacional é uma teoria matemática baseada nas relações entre conjuntos. Da sua aplicação ao Modelo Relacional de Banco de Dados, resultou a possibilidade de armazenar estruturas de dados complexas (como uma ficha cadastro de clientes), de maneira fragmentada, no formato tabular dos SGBR-R e recompor a informação original, a partir da formulação de relações entre as tabelas do banco de dados. Essas relações são providas pelos operadores da álgebra relacional, que se encontram disponíveis nos recursos da linguagem SQL (Structured Query Language). Os operadores da álgebra relacional classificam-se em dois grupos:

_ OPERADORES TRADICIONAIS DE CONJUNTO

- . UNIÃO
- . INTERSEÇÃO
- . DIFERENÇA
- . PRODUTO CARTESIANO

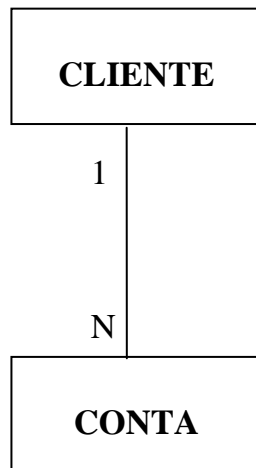
_ OPERADORES RELACIONAIS

- . PROJEÇÃO
- . SELEÇÃO
- . JUNÇÃO

Para classificar-se um SGBD como Relacional, é fundamental que ele possua, entre outras características, no mínimo os três operadores relacionais, haja vista que, nem todos os SGBD-R possuem os sete operadores. Os Operadores Tradicionais são mais encontrados em SGBD mais robustos, como ORACLE, SYBASE e DB/2.

1. ESTUDO DE CASO

O gráfico abaixo corresponde ao Modelo de Entidades e Relacionamentos (**MER**) de um banco de dados, que será utilizado como referência para o estudo dos operadores relacionais.



Segue uma amostragem das tabelas do banco de dados representado no MER:

CLIENTE

ID-CLI	NOME	ENDEREÇO	TIPO
001	RITA	SQN	V
002	MARCELO	GUARÁ	C
003	CARLA	GAMA	E
004	VÍTOR	SQS	C
005	RAQUEL	SQS	E
006	BRUNA	GUARÁ	V
007	SÔNIA	CRUZEIRO	C
008	GETÚLIO	SQN	C

C = COMUM
E= ESPECIAL
V= VIP

CONTA_CORRENTE

AGENCIA	NUM- CONT A	ID-CLI	SIT	SALDO
106	001	004	0	20.000,00
106	002	003	2	250,00
106	040	003	0	500,00
167	001	005	0	50,00
167	005	007	0	10,00
167	006	008	2	20,00
202	001	001	0	150,00
202	002	003	1	0
202	003	002	0	30,00
202	004	004	2	50.000,00

0 = ATIVA
1 = INATIVA
2 = BLOQUEADA

2. GENERALIDADES

- a. Nos SGBD que utilizam o SQL padrão ANSI (American National Standard Institute), os operadores da Álgebra Relacional são implementados por variações de parâmetros na sintaxe do comando SELECT, que é um comando de leitura da base de dados.
- b. A sintaxe utilizada para os comandos SELECT, que aparecerão nos exemplos, foi extraída dos manuais do SGBD ORACLE, que segue o padrão SQL ANSI. A estrutura básica do comando SELECT é:

```
SELECT colunas.... ou * (que significa todas as colunas)
FROM tabelas .....
WHERE condição.....
```

- c. As operações da álgebra relacional geram sempre uma tabela resultado residente em memória principal (tabela virtual), que em analogia com a teoria de conjuntos, pode ser vazia, unitária ou conter “N” linhas.
- d. As operações podem ser efetuadas entre duas tabelas virtuais através da combinação de dois comandos SELECT em uma única sentença.
- e. É comum a combinação de diversos operadores da Álgebra Relacional em um único comando “SELECT”. A análise individual de cada um deles é um exercício meramente didático.
- f. As situações criadas, são apenas ensaios, que não esgotam as possibilidades de utilização dos operadores. Além disso, uma mesma necessidade pode ter mais de uma solução. Portanto a utilidade dos operadores depende do problema tratado e da criatividade do técnico.

3. OPERADORES DE CONJUNTO

a. UNIÃO

A união de duas tabelas “A” e “B”, resulta numa tabela virtual “C”, contendo o total de linhas das tabelas envolvidas na operação.

No sistema exemplo, imagine que cada agência mantenha os dados cadastrais de CLIENTE em servidores locais de sua rede, e que esses servidores estão ligados em um servidor corporativo. Para se obter no servidor corporativo uma visão única, que contenha os dados de todos os clientes do Banco, pode-se utilizar o operador UNION da seguinte maneira:

```
SELECT * FROM cliente@agencia1;  
UNION  
SELECT * FROM cliente@agencia2;  
.  
.  
UNION  
SELECT * FROM cliente@agenciaN;
```

O resultado seria idêntico ao que temos na amostragem da tabela CLIENTE do sistema exemplo:

<u>ID-CLI</u>	<u>NOME</u>	<u>ENDERECO</u>	<u>TIPO</u>
001	RITA	SQN	V
002	MARCELO	GUARÁ	C
003	CARLA	GAMA	E
004	VÍTOR	SQS	C
005	RAQUEL	SQS	E
006	BRUNA	GUARÁ	V
007	SÔNIA	CRUZEIRO	C
008	GETÚLIO	SQN	C

Obs: Os SELECTs devem referenciar os mesmos atributos e na mesma seqüência.

b. INTERSEÇÃO

A Interseção entre duas tabelas “A” e “B”, resulta numa tabela virtual “C”, contendo as linhas comuns às duas tabelas envolvidas na operação.

No sistema exemplo, considere a necessidade de se listar o “ID-CLI” de clientes que possuam, simultaneamente, **CONTAS_CORRENTES ativas e bloqueadas**. Para atender a esse requerimento, pode-se utilizar o operador INTERSECT da seguinte maneira:

```
SELECT id_cli
  FROM conta_corrente
 WHERE sit = 0
INTERSECT
SELECT id_cli
  FROM conta_corrente
 WHERE sit = 2;
```

Considerando a amostragem da tabela CONTA_CORRENTE do sistema exemplo, o resultado do SQL anterior seria:

```
ID-CLI
  004
  003
```

Obs: Os SELECTs devem referenciar os mesmos atributos e na mesma seqüência.

c. DIFERENÇA

A Diferença entre duas tabelas “A” e “B” (na ordem A - B), resulta numa tabela virtual “C”, contendo as linhas pertencentes exclusivamente à tabela “A” e não a “B”.

No sistema exemplo, considere a necessidade de se listar o “ID-CLI” de clientes que não possuam contas_correntes INATIVAS ou BLOQUEADAS, somente ATIVAS. Para atender a esse requerimento, pode-se utilizar o operador MINUS da seguinte maneira:

```
SELECT id_cli  
FROM conta_corrente  
WHERE sit = 0  
MINUS  
SELECT id_cli  
FROM conta_corrente  
WHERE sit = 2 OR sit = 1;
```

Considerando a amostragem da tabela CONTA_CORRENTE do sistema exemplo, o resultado do SQL anterior seria:

```
ID-CLI  
005  
007  
001  
002
```

Obs: Os SELECTs devem referenciar os mesmos atributos e na mesma seqüência.

d. PRODUTO CARTESIANO

A Produto Cartesiano entre duas tabelas “A” x “B” resulta numa tabela virtual “C”, contendo todas as linhas da tabela “A” combinadas com todas as linhas da tabela “B”, através da concatenação de suas linhas.

Essa operação tem uma certa semelhança com a JUNÇÃO, pois combina dados de mais de uma tabela, exceto que não estabelece nenhum critério (join condition) para isso.

Geralmente o Produto é utilizado para construção de massas de teste ou quando o técnico esquece de colocar o “join condition” em um SELECT que envolva duas ou mais tabelas. Nesse caso, pode resultar numa tabela enorme. Por exemplo, o produto entre uma tabela “A” com 50 linhas e uma tabela “B” com 100 linhas resulta numa tabela “C” com 5.000 linhas.

O SELECT a seguir efetua um produto entre as tabelas CLIENTE e CONTA_CORRENTE:

```
SELECT nome, saldo  
FROM cliente, conta_corrente;
```

Considerando as amostragens das tabelas do sistema exemplo referenciadas no comando anterior, a tabela resultado teria 80 linhas, com o seguinte aspecto:

<u>NOME</u>	<u>SALDO</u>
RITA	20.000,00
RITA	250,00
RITA	500,00
.	.
.	.
MARCELO	20.000,00
.	.
MARCELO	50.000,00
.	.
.	.
SÔNIA	30,00
.	.
.	.
.	.
GETÚLIO	50.000,00

3. OPERADORES RELACIONAIS

e. PROJEÇÃO

A Projeção consiste em obter um subconjunto de colunas de uma ou mais tabelas_base, como resultado de uma consulta parcial aos dados disponíveis no banco de dados.

Geralmente é utilizada em conjunto com as demais operações para produzir resultados de consultas, ou ainda, para criar visões (VIEWS), que restringem o acesso do usuário a determinados atributos da base de dados.

No sistema exemplo, uma consulta contendo nome e endereço dos clientes, corresponde a uma PROJEÇÃO elaborada a partir da tabela-base CLIENTE, através da seguinte sentença SQL:

```
SELECT nome, endereço  
FROM cliente;
```

Considerando a amostragem da tabela CLIENTE do sistema exemplo, o resultado do SQL anterior seria:

NOME	ENDEREÇO
RITA	SQN
MARCELO	GUARÁ
CARLA	GAMA
VITOR	SQS
RAQUEL	SQS
BRUNO	GUARÁ
SÔNIA	CRUZEIRO
GETÚLIO	SQN

f. SELEÇÃO

Também conhecida como Restrição, essa operação tem por finalidade selecionar um subconjunto de linhas de uma ou mais tabelas_base, de acordo com critérios (where criteria), que envolvem atributos e valores para filtrar os dados desejados, gerando uma consulta parcial aos dados disponíveis no banco de dados.

Geralmente é utilizada em conjunto com as demais operações para produzir resultados de consultas, ou ainda, para criar visões (VIEWS), que restringem o acesso do usuário a determinadas linhas de tabelas na base de dados.

Os Critérios de Seleção são traduzidos na sintaxe do comando SELECT, pela combinação de operadores lógicos (AND, OR, NOT), aritméticos (=, <>, >, <, >= e <=) e operadores SQL (BETWEEN, LIKE, IN, NULL), representados na cláusula WHERE.

No sistema exemplo, uma consulta contendo somente os clientes VIP, corresponde a uma SELEÇÃO elaborada a partir da tabela-base CLIENTE, através da seguinte sentença SQL.

```
SELECT *  
FROM cliente  
WHERE tipo = 'V';
```

Considerando a amostragem da tabela CLIENTE do sistema exemplo, o resultado do SQL anterior seria:

ID-CLI	NOME	ENDEREÇO	TIPO
001	RITA	SQN	V
006	BRUNA	GUARÁ	V

g. JUNÇÃO

Essa operação relacional é utilizada para compor informações complexas a partir de tabelas relacionadas. A junção de duas tabelas “A” e “B” concatena as linhas das tabelas envolvidas, resultando numa tabela virtual “C”.

Para efetuar a JUNÇÃO de duas tabelas é essencial que elas estejam logicamente relacionadas, conforme prevê o modelo relacional, ou seja, o grau do relacionamento deve ser no máximo “1 : N”, sendo que a chave primária da entidade “1” deve figurar como chave estrangeira da entidade “N”. Além disso, os valores dessas chaves devem ser coincidentes, para as linhas que se deseja concatenar.

A junção é notada na sintaxe do SQL, pela comparação de atributos chave primária / chave estrangeira, através da cláusula WHERE do comando SELECT, o que denominamos condição de junção (join condition). Quando o técnico esquece de colocar o “join condition” em um SELECT que envolva duas ou mais tabelas o SGBD geralmente efetua o PRODUTO.

O SELECT a seguir efetua uma junção entre as tabelas CLIENTE e CONTA_CORRENTE:

```
SELECT nome, saldo  
FROM cliente, conta_corrente  
WHERE cliente.id-cli = conta_corrente.id-cli;
```

Considerando as amostragens das tabelas do sistema exemplo referenciadas no comando anterior, a tabela resultado seria:

NOME	SALDO
RITA	150,00
MARCELO	30,00
CARLA	500,00
CARLA	0,00
VITOR	20.000,00
VITOR	50.000,00
RAQUEL	50,00
SÔNIA	10,00
GETÚLIO	20,00

Note que a cliente de nome BRUNA não figura na tabela resultado porque não possui registro na tabela CONTA_CORRENTE.

PARTE II - PROJETO DE BANCO DE DADOS

INTRODUÇÃO

A modelagem de um sistema de processamento de dados pode ser feita a partir de dois enfoques:

- 1º) Enfoque dos DADOS que são processados;
- 2º) Enfoque das FUNÇÕES que tratam esses dados.

Esses dois enfoques são complementares e, usados conjuntamente, fornecem ao analista os dois principais ângulos do problema em análise: DADOS E FUNÇÕES. O Diagrama de fluxo de dados (DFD) e o Diagrama Hierárquico Funcional (DHF) são exemplos de ferramentas com enfoque FUNCIONAL e o Modelo de Entidade e Relacionamentos (MER) é o diagrama utilizado para prover a visão dos DADOS.

Os modelos Funcional e de Dados, ao integrarem um projeto de sistemas, constituem-se em importantes instrumentos para, entre outras coisas:

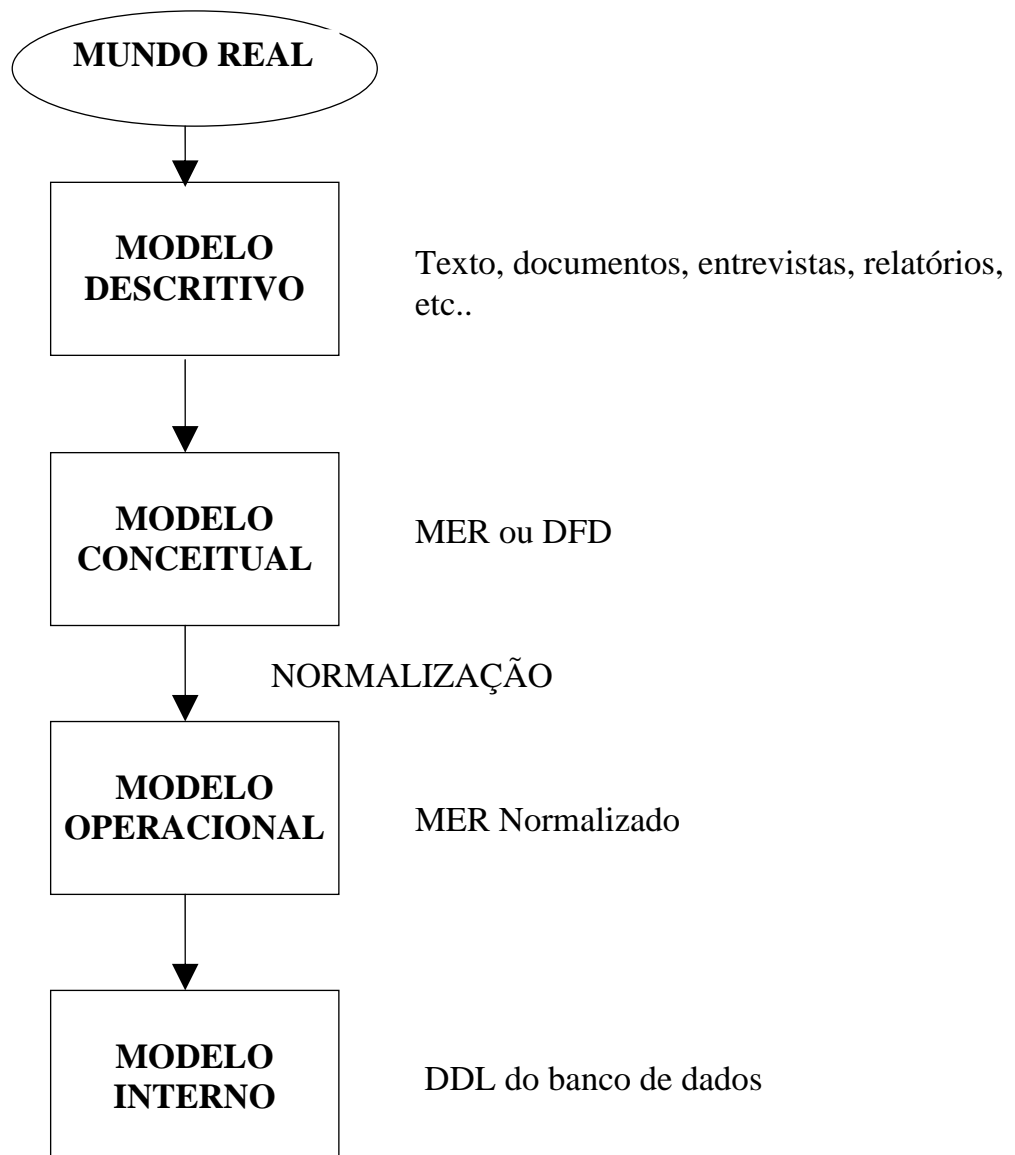
- _ Apresentar, através de diagramas, uma síntese do problema, destacando seus aspectos mais relevantes;
- _ Oferecer uma visão contextual do sistema em análise.
- _ Facilitar a comunicação entre os integrantes da equipe de desenvolvimento (gerentes, analistas, usuários, etc);
- _ Motivar a participação do usuário no projeto;
- _ Documentar o processo de análise, para que o trabalho não sofra solução de continuidade;

Apesar dos dois enfoques (Funcional e de Dados) do processo de análise de sistemas, este texto trata, especificamente, da MODELAGEM DE DADOS, com o emprego do MODELO DE ENTIDADES E RELACIONAMENTOS e da técnica de NORMALIZAÇÃO, adotando a METODOLOGIA DE PROJETO DESCENDENTE (Setzer, Valdemar) como referencial metodológico para o processo de modelagem.

CAPÍTULO VIII

NÍVEIS DE ABSTRAÇÃO

Ao projetar um sistema de informações para ser implantado no computador, o projetista elabora um modelo da realidade, visando adequá-la às limitações do ambiente do computador. Devido à complexidade do mundo real e seguindo a linha de abordagem “TOP DOWN”, o processo de modelagem atravessa diversas fases, às quais Setzer¹ denominou NÍVEIS DE ABSTRAÇÃO, em sua Metodologia de Projeto Descendente, que usaremos como referência neste texto e que encontra esquematizada na figura a seguir:



¹ Waldemar W. setzer - livro:Projeto de Banco de Dados

Os NÍVEIS DE ABSTRAÇÃO, propostos por Setzer, representam as diferentes visões que um analista obtém de uma realidade (MUNDO REAL), a medida que avança no processo de modelagem, até chegar ao sistema implantado. A partir do **MUNDO REAL**, Setzer propõe quatro níveis de abstração, que são os modelos **DESCRITIVO, CONCEITUAL, OPERACIONAL E INTERNO**.

1. MUNDO REAL

O MUNDO REAL envolve todos os objetos (normas, pessoas, eventos, fatos, organismos sociais, etc.), que compõem o CENÁRIO, do qual, o analista deverá extrair a parte a ser representada no computador, que pode ser uma empresa, um departamento, setor, processo de negócio, etc..

2. MODELO DESCRITIVO

Este modelo é representado por um texto contendo a descrição do mundo real (ambiente alvo do estudo), explicitando as características do problema a ser tratado através de regras do negócio, elenco de necessidades, a análise de requisitos, questionários, formulários, relatórios, informações sobre volume, etc.. Constitui o primeiro nível da modelagem, onde o analista estabelece as fronteiras do sistema.

Neste nível de abstração, o modelador identifica a área alvo da modelagem, descreve o problema a ser solucionado e identifica os componentes sobre os quais se deseja armazenar e recuperar informações. Deve-se relatar o maior o nível de detalhes possível a respeito do problema, cuidando-se para não perder a objetividade.

O maior problema do modelo descritivo é a falta de padrão. Existem várias maneiras de se descrever o mesmo problema, as palavras possuem significados variados e podem causar dupla interpretação, ou seja, ambigüidade. Para amenizar esse problema, deve-se evitar períodos longos, procurar trabalhar com tópicos, frases curtas e utilizar palavras precisas e adequadas ao vocabulário profissional do usuário.

3. MODELO CONCEITUAL

É um modelo baseado em símbolos PADRONIZADOS, que expressa, graficamente, os CONCEITOS do MUNDO REAL delineados no Modelo Descritivo. A utilização de uma simbologia padronizada torna a linguagem mais precisa, o que facilita a comunicação entre os integrantes da equipe de desenvolvimento, especialmente, entre analistas e usuários. Portanto, esse modelo é o mais adequado para o registro e validação dos conceitos do mundo real, nas fases do projeto que envolvem o usuário.

O produto dessa fase da análise deve ser um MODELO DE ENTIDADES E RELACIONAMENTOS (MER) enfocando a visão dos DADOS com alto nível de abstração, ou seja, voltado para o MUNDO REAL e sem preocupar-se com detalhes do ambiente operacional, como, por exemplo, o software e hardware no qual o sistema será implantado.

Neste nível da modelagem, deve-se ressaltar os aspectos mais relevantes do problema, ou seja, representa-se principais entidades, relacionamentos, estruturas de dados e integração com outros sistemas. Os detalhes do banco de dados (chaves primárias, estrangeiras, integridades, etc.) devem ser deixados para a fase seguinte, de construção do modelo operacional.

4. MODELO OPERACIONAL

O modelo Operacional decorre da análise do modelo Conceitual, visando torna-lo adequado ao ambiente operacional, no qual o sistema será implantado. No caso específico deste texto, essa análise visa adaptar o modelo Conceitual às limitações de um Sistema Gerenciador de Banco de Dados Relacional (SGBD-R).

Obter o projeto das tabelas do banco de dados é o principal objetivo dessa fase e para alcançá-lo, o analista deve investigar cada entidade do Modelo Conceitual, decompor estruturas em elementos de dados, utilizar a NORMALIZAÇÃO e o conhecimento sobre o MODELO RELACIONAL como balizadores de suas ações.

Nesta fase, o dicionário de dados do Modelo Conceitual é enriquecido com a escolha das chaves primárias, definição de chaves estrangeiras, índices, integridades, e outros detalhes essenciais para a criação do banco de dados.

5. MODELO INTERNO

Corresponde ao sistema implementado no computador, ou seja, é representado pelo conjunto de linhas de código geradas na linguagem do SGBD, que no caso dos modelos Relacionais é o SQL (structured query language).

Do ponto de vista teórico, o modelo Operacional seria o ideal para ser implantado. Porém, a prática revela que esse modelo ainda pode sofrer modificações antes que o BD seja criado. Os principais fatores que determinam essas modificações são a **MELHORIA DE PERFORMANCE** e a **DISTRIBUIÇÃO DE DADOS**.

Visando aumentar a performance, podem ser criadas redundâncias, com a desnormalização e criação de tabelas totalizadoras para geração de informações gerenciais. Nesses casos, deve-se ter especial cuidado com a consistência do BD no processo de atualização dos dados. Além disso, deve-se considerar que a redundância de dados aumenta os custos com processamento e armazenamento de dados, que devem ser compensados pelo benefício esperado.

Com relação à distribuição de dados é comum a criação de cópias de tabelas (distribuição por replicação) ou a fragmentação de uma entidade do modelo operacional em várias tabelas não redundantes (distribuição por particionamento). Em ambos os casos, as partes resultantes são distribuídas em diversos nós da rede e a manutenção dessas partes será facilitada caso o SGBD possua algum mecanismo de gerenciamento automático de distribuição de dados.

CAPITULO IX

MODELO DE ENTIDADES E RELACIONAMENTOS

O MODELO DE ENTIDADE E RELACIONAMENTOS (MER), foi proposto originalmente por PETER CHEN em 1976. Seu objetivo, ao publicar esse trabalho, era consagrar um método eficiente para representar os dados e ressaltar a diferença entre as estruturas suportadas pelos SGBD HIERÁRQUICOS, REDE e RELACIONAL. Atualmente o MER tem sido utilizado para representar a visão dos dados no projeto de banco de dados.

A principal vantagem do MER é a simplicidade. O Modelo de Entidades e Relacionamentos possui apenas três componentes básicos: ENTIDADE, ATRIBUTO e RELACIONAMENTO (e seus respectivos símbolos para diagramação).

A proposta original de CHEN foi enriquecida por trabalhos de diversos autores, que procuraram atribuir maior capacidade semântica ao MER, o que gerou as camadas EXTENSÕES. As extensões ao MER possibilitam representar o mundo real com maior riqueza de detalhes. Por outro lado, elas provocaram a diversificação dos padrões de notação e terminologia, contrastando com a proposta original de simplicidade e acarretando problemas para a disseminação da cultura.

Este capítulo procura exprimir uma visão objetiva do Modelo de Entidades, para instrumentalizar o analista frente à necessidade de representar a visão dos dados de um projeto de sistema.

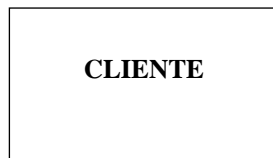
1. NOTAÇÃO E TERMINOLOGIA

ENTIDADE é a representação genérica de um COMPONENTE DO MUNDO REAL, SOBRE O QUAL DESEJAMOS ARMAZENAR INFORMAÇÕES (ATRIBUTOS).

As ENTIDADES podem representar coisas tangíveis (pessoal, material, patrimônio, etc.) ou intangíveis (eventos, conceitos, planos, etc.).

Para NOTAR graficamente uma entidade emprega-se um RETÂNGULO identificado por um substantivo (simples ou composto).

Exemplo:



2. REGRAS PARA ATRIBUIÇÃO DE NOMES À ENTIDADES

A literatura não consagra um padrão para a atribuição de nomes a entidades, mas, para alcançarmos um mínimo de padronização, indicamos observar as seguintes regras:

- Nomes breves e objetivos, grafados em maiúsculas e, que identifiquem facilmente o conteúdo da entidade;
- No singular, já que a pluralidade decorre, naturalmente, do número de ocorrências (linhas / tuplas), característica própria de toda entidade.
- Nomes compostos separados por hífen, eliminando-se o uso de preposições ou outros termos de ligação.
- Evitar abreviação de nomes. Se necessário, ampliar o tamanho da figura representativa da entidade.

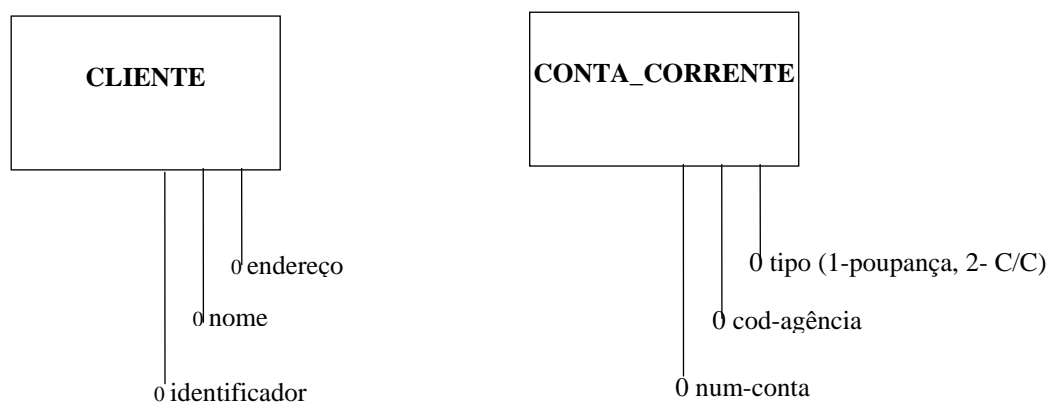
3. ATRIBUTO

Os atributos são os dados que devemos armazenar a respeito da entidade, para atender às necessidades de informações demandadas pelo usuário. Constituem tudo o que se pode relacionar como próprio (propriedade) da entidade e que, de alguma forma, estejam contidos no escopo do problema em análise. Os atributos qualificam e distinguem as entidades no MER.

Em relação ao banco de dados, os atributos representam as colunas, que formam a estrutura de dados das tabelas. As colunas armazenam um valor para cada linha. Esse valor armazenado é designado por VALOR DE ATRIBUTO.

O conjunto de valores de atributos, distintos por um identificador único (chave primária) denomina-se OCORRÊNCIA. Esse conceito é análogo ao de linha (tupla) em tabela relacional e de registro em arquivo convencional.

Pode-se exprimir os atributos no MER, conforme mostrado na figura abaixo:



Cabe observar, que a representação de atributos no MER pode “poluir” o gráfico, comprometendo sua objetividade e visão contextual. Esse recurso deve ser reservado para situações especiais, em que você queira destacar um atributo, por considera-lo elucidativo para o contexto.

4. DICIONÁRIO DE DADOS

Os ATRIBUTOS são usualmente descritos sob a forma de estruturas e elementos no Dicionário de Dados, onde deve constar uma relação de atributos para cada entidade do MER. As notações mais utilizadas para criação de dicionários de dados são as definidas por GANE² e YOURDON³.

Exemplo da notação de Gane para descrição de estruturas de dados:

CLIENTE

IDENTIFICADOR
NOME
ENDEREÇO

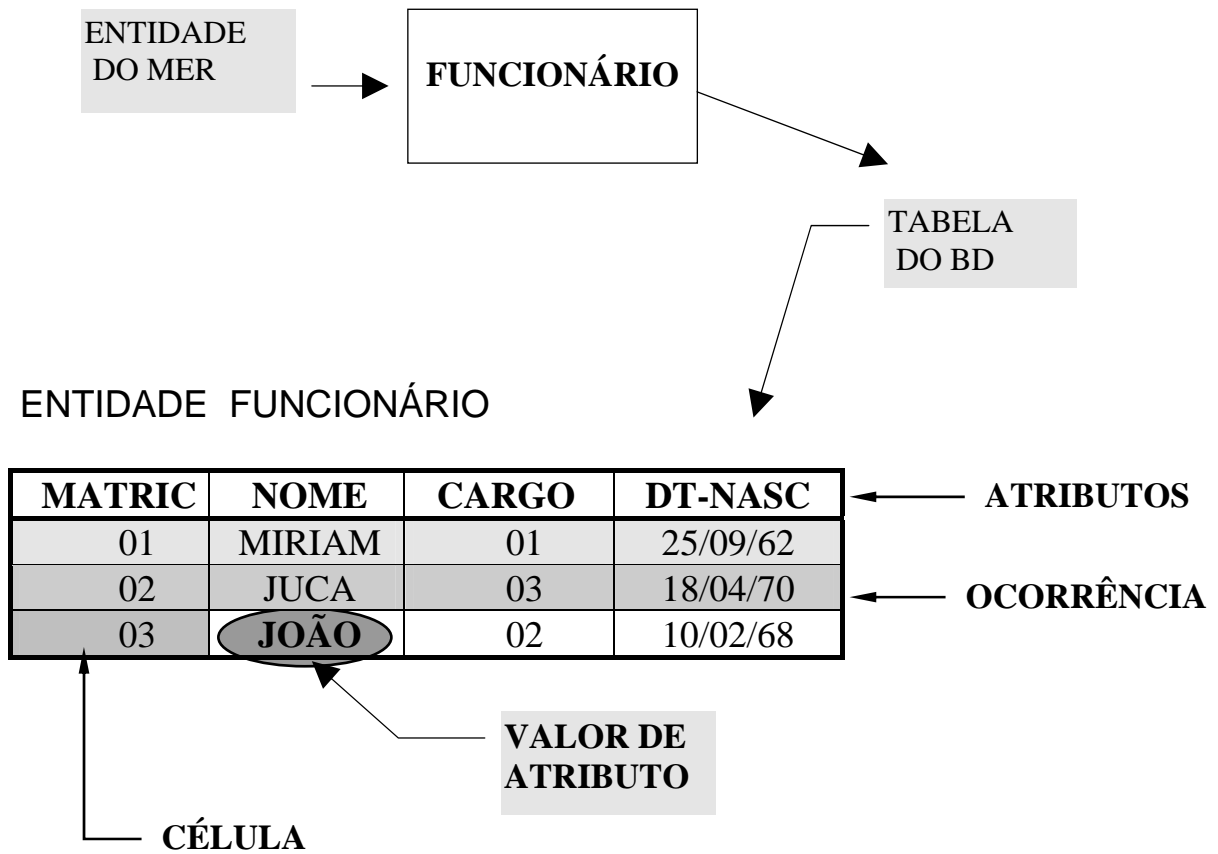
CONTA_CORRENTE

NUMERO_CONTA
TIPO-CONTA (1-poupança, 2-c/c)
AGÊNCIA
 CÓDIGO_AGÊNCIA
 ENDEREÇO_AGÊNCIA
LANÇAMENTOS*
 NUMERO_LANÇAMENTO
 DATA
 TIPO (deb, cre)
 VALOR

² GANE, CHRIS - LIVRO: ANÁLISE ESTRUTURADA DE SISTEMAS

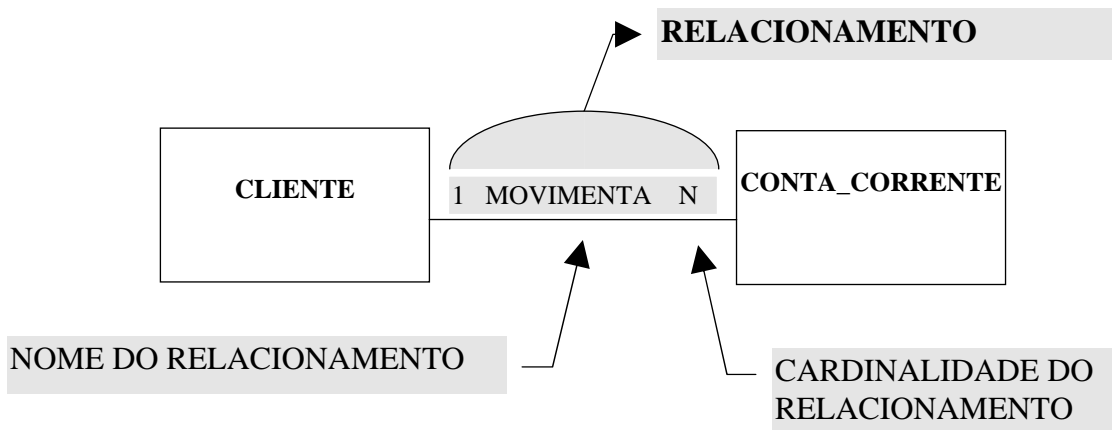
³ YOURDON, EDWARD - LIVRO: ANÁLISE ESTRUTURADA MODERNA

A figura a seguir ilustra conceitos vistos nesse capítulo, representando a entidade “FUNCIONÁRIO” sob a forma de uma tabela do banco de dados.

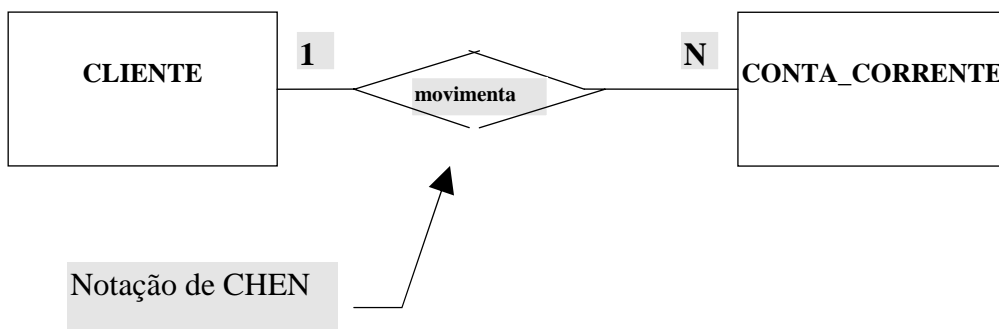


5. RELACIONAMENTO

O relacionamento representa a relação existente entre entidades integrantes de um MER. É notado por uma LINHA ligando as ENTIDADES envolvidas e possuem NOME e CARDINALIDADE. Veja a figura a seguir.



Peter Chen⁴ utiliza um LOSANGO para representar o RELACIONAMENTO entre entidades, porém, a experiência demonstra que o uso dessa notação contribui para “poluir” o gráfico e não produz resultado prático, exceto em casos de relacionamentos que envolvam mais de duas entidades (relacionamento múltiplo - modelo conceitual).



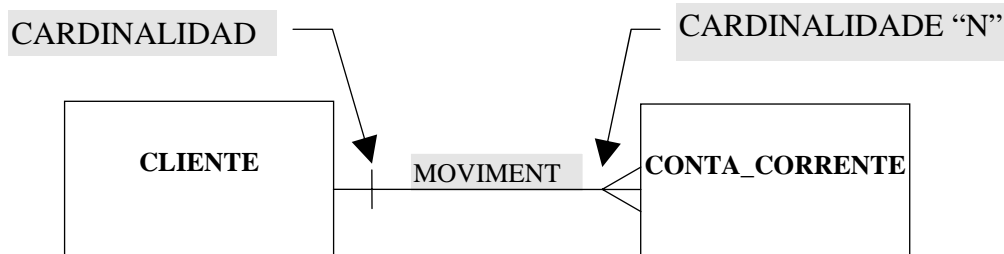
⁴ CHEN, PETER - LIVRO: MODELO DE ENTIDADES E RELACIONAMENTOS

6. CARDINALIDADE (GRAU DO RELACIONAMENTO)

A cardinalidade constitui um indicativo genérico da quantidade de ocorrências (máxima e mínima) de cada ENTIDADE envolvida no relacionamento. É expressa por sinais (flechas, pés-de-galinha, números, letras, etc..), que são grafados sobre a linha do relacionamento, nas duas extremidades do mesmo.

A notação para a cardinalidade é o item que apresenta maior variação entre os autores que escrevem sobre o MER. Neste texto usaremos uma barra para notar a cardinalidade "1" e o pé-de-galinha para a cardinalidade "N".

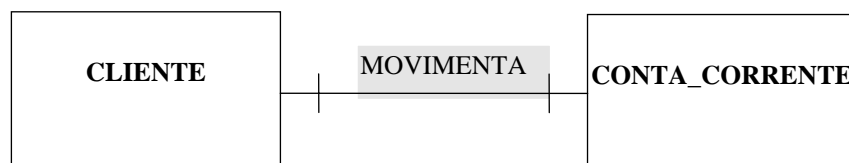
Exemplo:



Considerando a cardinalidade, o relacionamento pode ser de três tipos:

1º) 1:1 - Lê-se UM para UM

Exemplo:



Indica que UMA ocorrência da entidade CLIENTE relaciona-se com UMA ocorrência da entidade CONTA-CORRENTE e vice-versa,

2º) 1:N - Lê-se UM para MUITOS

Exemplo:



Indica que UMA ocorrência da entidade CLIENTE relaciona-se com muitas ocorrências da entidade CONTA-CORRENTE e vice-versa,

3º) M:N - Lê-se MUITOS para MUITOS

exemplo:



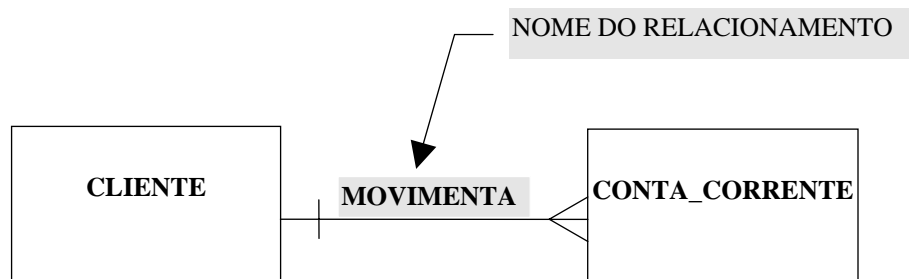
Indica que:

- UMA ocorrência da entidade CLIENTE relaciona-se com MUITAS ocorrências da entidade CONTA- CORRENTE e;
- UMA ocorrência da entidade CONTA-CORRENTE relaciona-se com MUITAS ocorrências da entidade CLIENTE. (cada das contas conjuntas)

7. NOME DO RELACIONAMENTO

É o componente do modelo E-R que identifica o relacionamento, justificando e esclarecendo a importância de sua existência para o contexto estudado.

Exemplo:



Nos casos onde o relacionamento é óbvio torna-se dispensável a atribuição de nome ao mesmo.

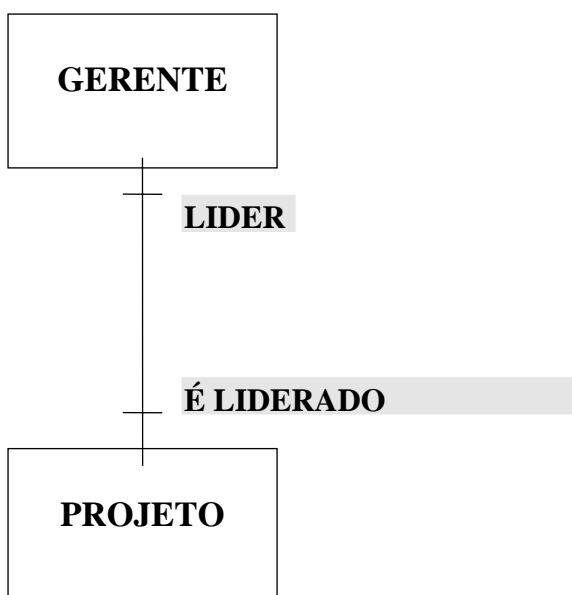
O nome do relacionamento é recomendável nas seguintes situações:

- Quando existirem diversas possibilidades óbvias de relacionamentos entre o par de entidades, sendo que, deseja-se representar apenas em;
- Por questão documental, para dar maior clareza ao modelo;
- Caso ocorra mais do que um relacionamento entre o par de entidades (relacionamento duplo, triplo, etc.);
- No caso de AUTO-RELACIONAMENTOS (entidade relacionando-se com ela mesma- recursividade)
- Quando da utilização do MER para representar modelos a serem implementados em SGBD REDE
- Quando da utilização de CASE para desenho do MER (caso a ferramenta obrigue)

8. PAPEL

O papel das entidades no relacionamento fica implícito no nome e na cardinalidade do mesmo e pode ser inferido a partir desses componentes. Porém, especificar o PAPEL que cada entidade desempenha é uma alternativa, que pode substituir, com maior precisão, a colocação do nome no relacionamento, atribuindo ao MER maior capacidade semântica.

EXEMPLO:



OBS: A maioria das ferramentas CASE não utilizam o PAPEL como alternativa para a construção de MER.

9. SENTENÇAS (REGRAS DO NEGÓCIO)

Duas sentenças são derivadas da leitura do relacionamento. Elas refletem as regras do negócio e ajudam na validação do modelo junto ao usuário.

Exemplo:



_ Sentença-1:

UM CLIENTE é titular de **VÁRIAS** CONTAS-CORRENTES.



_ Sentença-2:

UMA CONTA-CORRENTE tem como titular **UM** CLIENTE.



OBS: As sentenças sempre começam com **UM / UMA** e a cardinalidade considerada é a do lado oposto ao da primeira entidade citada na sentença.

10. INTEGRIDADE REFERENCIAL (IR)

A Integridade Referencial é notada no MER através de sinalização colocada no relacionamento junto à marca de cardinalidade, que indica se o relacionamento é OBRIGATÓRIO ou OPCIONAL (TOTAL / PARCIAL). Os sinais utilizados para notar a IR, variam muito, conforme os autores ou ferramenta CASE adotados e se confundem com a marca de cardinalidade.

Veja no quadro a seguir as notações mais utilizadas para IR:

OPCIONAL	OBRIGATÓRIO	AUTOR
○ (bolinha aberta)	● (bolinha fechada)	JAMES MARTIN
sem marcação	(uma barra vertical)	DIVERSOS
(uma barra vertical)	(duas barras verticais)	DIVERSOS
linha do relacionamento tracejada	linha cheia no relacionamento	BACHMAN / GANE

Exemplo:



11. REGRAS GERAIS

- Não podem existir duas entidades iguais no mesmo modelo, ou seja, que representem o mesmo objeto do mundo real e possuam os mesmos atributos; mesmo que elas possuam nomes diferentes.
- Cada entidade deve possuir pelo menos dois atributos (colunas) e duas ocorrências (linhas).
- _ No Modelo Operacional os relacionamentos devem envolver, no máximo, duas entidades.
- Os relacionamentos Múltiplos do Modelo Conceitual devem ser transformados em entidades em sua passagem para o modelo operacional.
- Pode existir mais do que um relacionamento entre o mesmo par de entidades (relacionamento duplo, triplo, etc..)
- Para cada relacionamento com cardinalidade “N:N” do Modelo Conceitual, deve ser criada, no Modelo Operacional, uma ENTIDADE ASSOCIATIVA. Essa entidade será ligada às demais por dois relacionamentos “1:N”, sendo que as cardinalidades “N” de cada relacionamento, serão marcadas ao seu lado.
- Deve-se avaliar os relacionamentos com cardinalidade “1:1”, verificando se o par de entidades envolvidas pode ser representado por uma entidade única.
- Cada entidade do MER deve participar de pelo menos um relacionamento. Caso isso não ocorra é provável que a entidade isolada não faça parte do contexto modelado.

12. VANTAGENS E DESVANTAGENS DO MER

Vantagens:

- Simplicidade da notação e terminologia
- Rápida absorção dos conceitos por parte dos técnicos
- Facilidade de compreensão por parte dos usuários
- Grande possibilidade de validação do modelo por parte do usuário
- Capacidade de representar diversos níveis de abstração
- Compreensão mais objetiva, mais formal e portanto menos ambígua da do problema.

Desvantagens do MER:

- Diversidade da notação e terminologia
- Nenhuma ênfase aos processos que manipulam as informações.

CAPITULO X

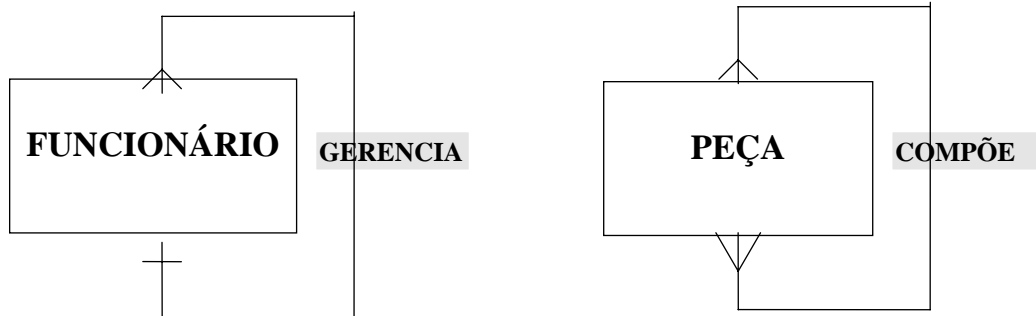
TIPOS ESPECIAIS DE RELACIONAMENTO

Os casos apresentados a seguir representam situações especiais que ocorrem esporadicamente. Geralmente os relacionamentos do MER são do tipo 1:N entre um par de entidades.

1. AUTO-RELACIONAMENTO

O auto-relacionamento representa a relação entre linhas de uma mesma tabela. É também conhecido como RECURSIVIDADE.

Exemplos:

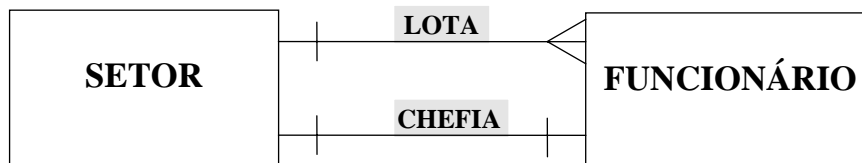


UM funcionário gerencia "N" funcionários
UM funcionário é gerenciado por "1" funcionário

UMA peça compõe "N" peças
UMA peça é componente de "N" peças

2. MAIS DE UM RELACIONAMENTO (duplo, triplo, etc.)

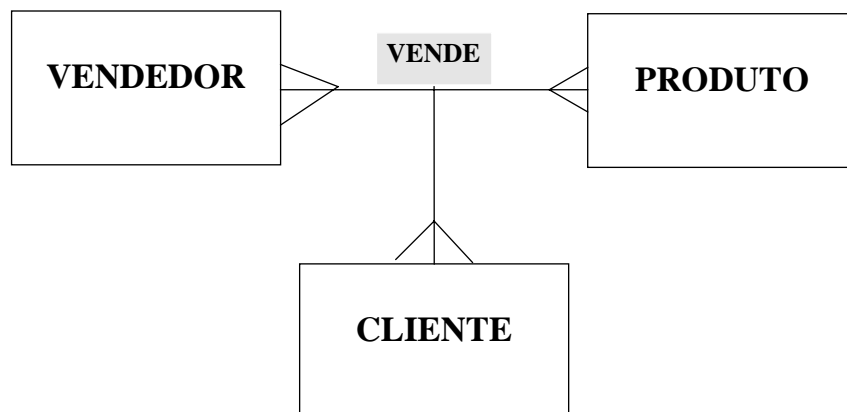
Ocasionalmente pode ser necessário representar mais de um relacionamento entre o mesmo par de entidades. Esses relacionamentos serão válidos desde que denotem situações distintas e independentes. Neste caso, existirão no Modelo Operacional, tantas chaves estrangeiras quantos forem os relacionamentos.



3. RELACIONAMENTO MÚLTIPLO

O Relacionamento Múltiplo, previsto por PETER CHEN, é aquele que relaciona mais do que duas entidades. Este tipo de relacionamento somente ocorre em Modelos Conceituais e geralmente denota eventos. Para implementá-lo nos SGBD-R é necessário criar uma entidade, em procedimento análogo ao que é adotado nos casos de relacionamento “N:N”.

Exemplo:



CAPITULO XI

EXTENSÕES AO MER

EXTENSÕES AO MER são conceitos e simbologias criados por diversos autores que escrevem sobre o Modelo de Entidades, para representar situações específicas, não cobertas pela proposta original de PETER CHEN. As extensões possibilitam ao modelador tornar o modelo mais genérico (conceitual) ou mais específico (operacional), conforme a necessidade. Neste capítulo destacamos algumas extensões as quais julgamos importantes para a abordagem metodológica que adotamos neste texto.

As extensões que abordaremos são as seguintes:

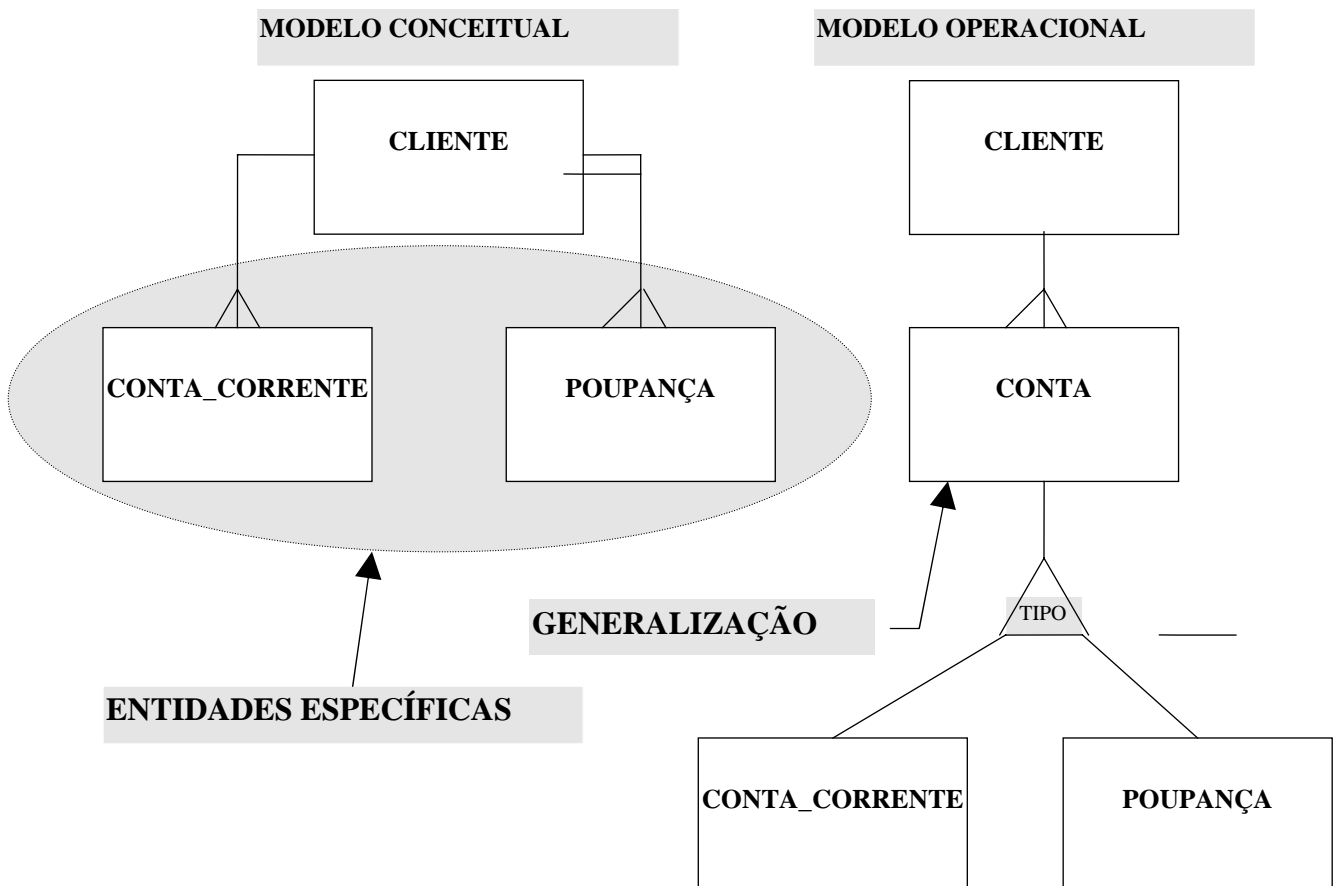
- _ Generalização (Setzer e outros) ou Supertipo (Gane)
- _ Especialização (Setzer e Outros), Subtipo (Gane), particionamento ou classificação (outros).
- _ Agregação (Setzer)

1. GENERALIZAÇÃO

É o resultado da união de dois ou mais conjuntos de entidades afins, de mais baixo nível de abstração, produzindo uma entidade mais genérica. Portanto, a generalização parte de entidades **ESPECÍFICAS** para criar uma **GENÉRICA**.

O exemplo a seguir apresenta dois tipos de conta (corrente e poupança). No Modelo Conceitual elas apresentam-se como entidades independentes. Verificado que elas possuem atributos em comum, criou-se no Modelo Operacional a entidade genérica "CONTA", que contém os atributos comuns aos dois tipos de conta, além de um atributo "TIPO" para diferenciá-las. Essa solução pode simplificar o processo de atualização e facilitar determinadas consultas que manipulem informações gerais sobre os dois tipos de conta.

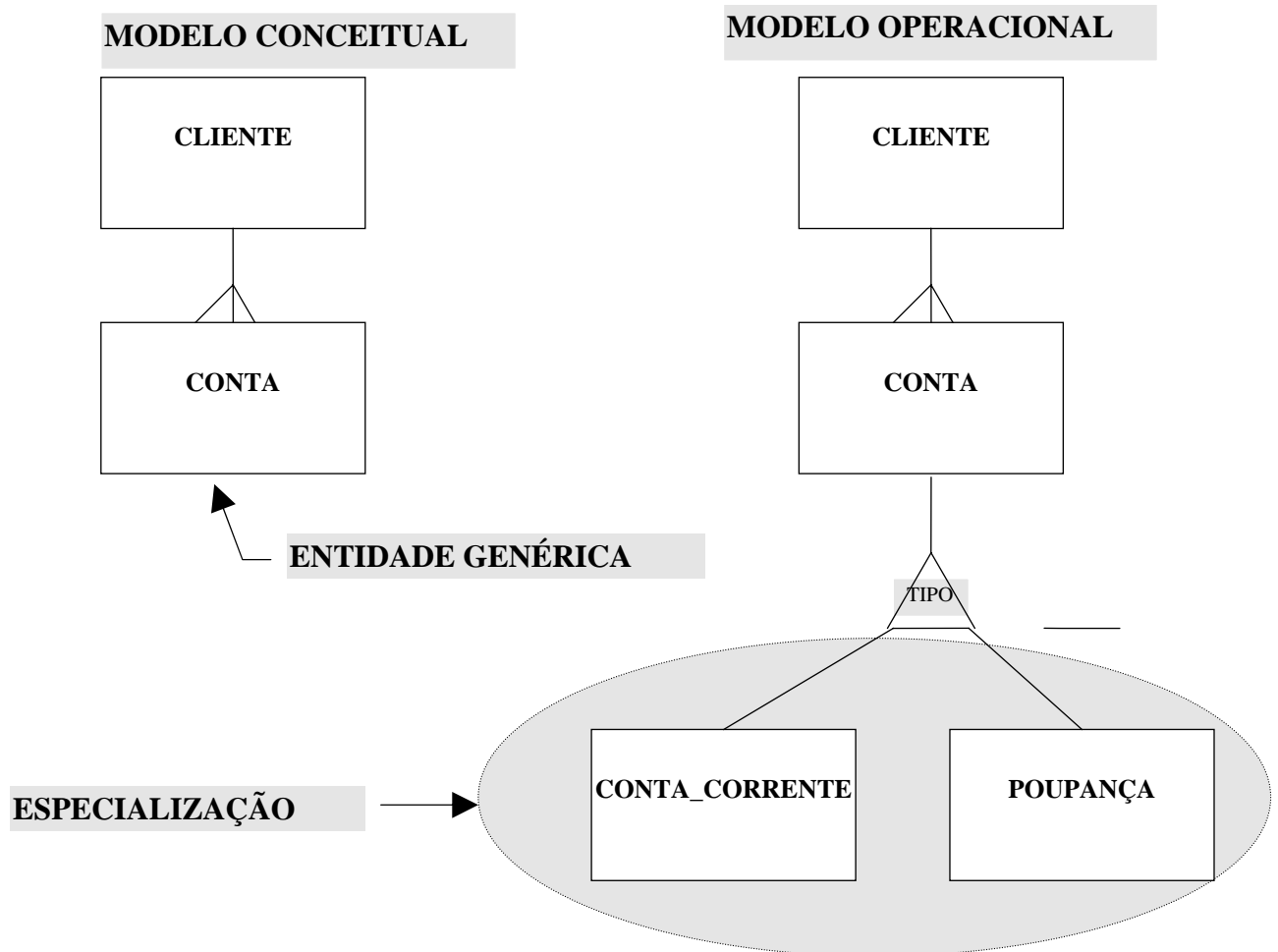
Exemplo:



2. ESPECIALIZAÇÃO

É o resultado da divisão de uma entidade genérica, em subconjuntos de entidades, que contenham atributos específicos de determinadas categorias de entidades. Portanto, a Especialização parte de uma entidade **GENÉRICA** para criar **ESPECÍFICAS**.

Exemplo:

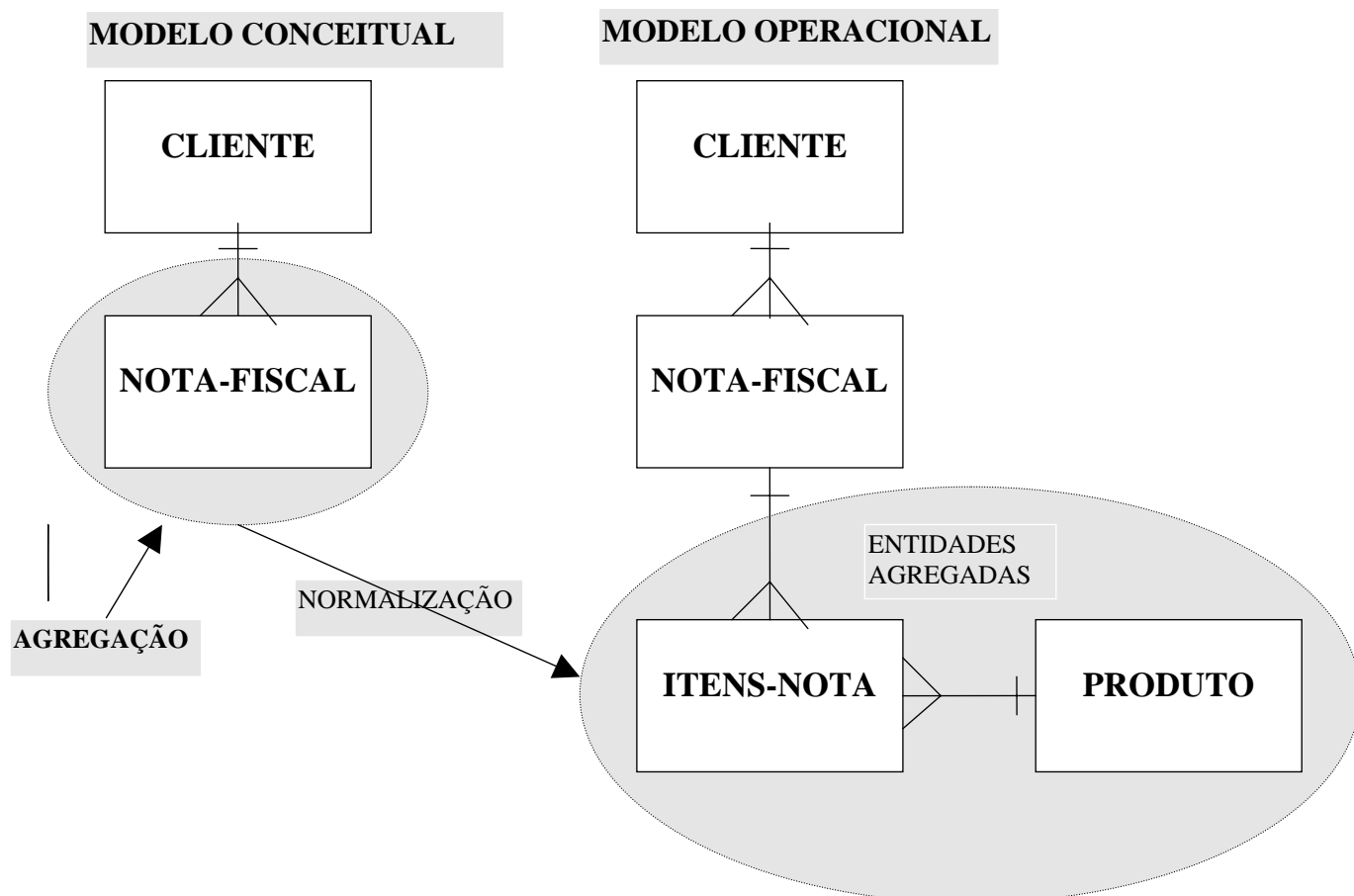


3. AGREGAÇÃO

Na abordagem “top-down” (do geral para o específico) é comum representar-se no Modelo Conceitual apenas as entidades mais importantes para a elucidação do problema. Assim, uma única entidade do Conceitual, ao ser normalizada, pode dar origem a um conjunto de entidades e seus relacionamentos no Modelo Operacional.

Portanto, a **AGREGAÇÃO** é um recurso do Modelo Conceitual, que representa através de uma única entidade “não-normalizada”, um conjunto de entidades e relacionamentos do nível Operacional.

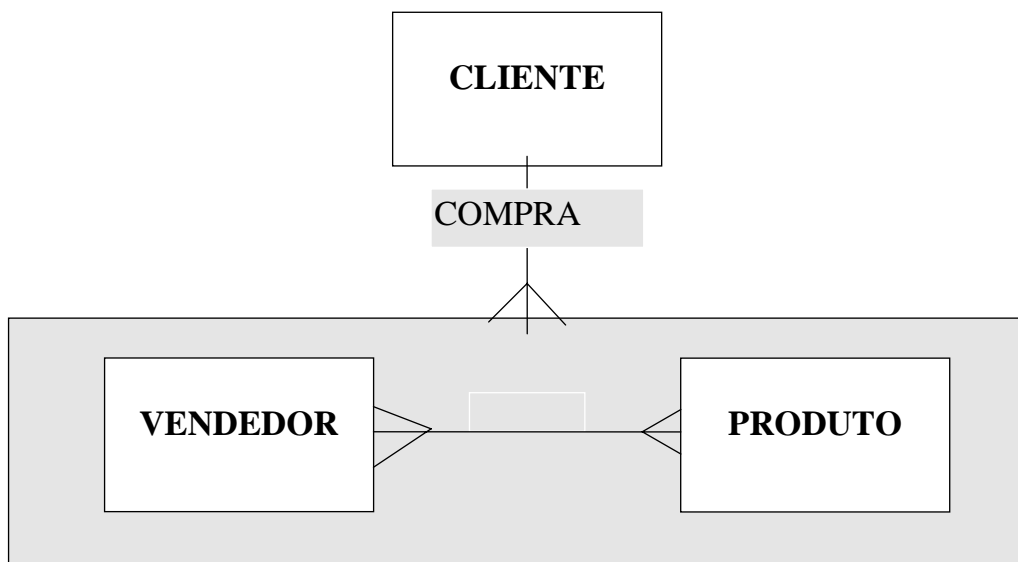
A Agregação permite visualizar um modelo complexo, com muitas entidades, de uma maneira mais genérica. Com isso, obtém-se uma melhor visão do contexto, com destaque para as entidades e relacionamentos mais importantes e omissão dos detalhes irrelevantes.



4. VARIAÇÃO DO CONCEITO DE AGREGAÇÃO

Alguns autores não admitem o relacionamento múltiplo e utilizam a Agregação para representar o relacionamento de uma entidade com um par de entidades, conforme mostrado no gráfico a seguir. Neste texto não adotaremos essa linha de pensamento.

Exemplo:



CAPÍTULO XII

NORMALIZAÇÃO

A NORMALIZAÇÃO é uma técnica de modelagem de dados, criada por E. F. CODD, nos laboratórios de pesquisa da IBM, lançada junto com as bases do modelo Relacional de SGBD. Essa técnica de modelagem nos proporciona critérios objetivos, para determinarmos quando uma relação (tabela / estrutura de dados) apresenta problemas no tocante à observância de princípios do enfoque relacional, tais como:

- Tabela bidimensional (valores atômicos)
- Regras de integridade
- Mínima redundância
- Nenhuma inconsistência
- Inexistência de anomalias de atualização (inclusão, alteração e exclusão)

O processo de NORMALIZAÇÃO proposto por CODD, deu origem a três FORMAS NORMAIS:

- _ PRIMEIRA FORMA NORMAL - 1FN;
- _ SEGUNDA FORMA NORMAL 2FN e;
- _ TERCEIRA 3FN.

Outras formas normais foram propostas, por diversos autores, configurando situações que ocorrem mais raramente, sendo a 4FN a mais significativa.

Conforme veremos mais adiante, a 1FN visa tão somente colocar as estruturas de dados oriundas dos modelos conceituais no formato tabular adequado, que permita que elas possam ser criadas nos SGBD-R. Nesse sentido, considera-se que relações em 1FN já estão NORMALIZADAS.

As demais formas normais estão dirigidas para evitar REDUNDÂNCIA DE DADOS, INCONSISTÊNCIAS e ANOMALIAS DE ATUALIZAÇÃO. Redundância de dados é um fato gerador de inconsistências, já as anomalias de atualização criam dificuldades operacionais para a manutenção do BD. Esses aspectos reforçam a importância de aplicação da 2FN e 3FN.

1. ANOMALIAS DE ATUALIZAÇÃO

São problemas presentes em estruturas de dados modeladas de forma inadequada.

TABELA FUNCIONÁRIO

MATR	NOME	ENDEREÇO	COD-ORGÃO	SIGLA-ORG	QTD-FUNC
03	JOÃO	SQS	01	GETAE	2
05	JOSÉ	SQS	01	GETAE	2
01	VILMA	GAMA	05	GEPAC	1
02	ANA	GUARA	02	GEPRO	3
08	JUCA	SQN	02	GEPRO	3
06	ANA	SQN	02	GEPRO	3
.
.
.

Exemplos de anomalias de atualização na tabela FUNCIONÁRIOS:

- A **INCLUSÃO** de um novo **ORGÃO** na tabela fica condicionada a que algum funcionário seja alocado nele;
- A **ALTERAÇÃO** de nome do órgão “GERAE” para “GETAE” provoca atualização em várias tuplas, haja vista, que o mesmo pode repetir-se numerosas vezes na relação;
- A **INCLUSÃO** de um novo funcionário para o “GEORG” causa **ALTERAÇÃO** no atributo “QT-FUNC” em diversas tuplas;
- A **EXCLUSÃO** da funcionária “VILMA” da tabela ocasiona perda de informações sobre o ‘GEPAC”;

2. TERMINOLOGIA

O vocabulário de NORMALIZAÇÃO se confunde com o empregado nos SGBD do modelo RELACIONAL. Isso ocorre por que a técnica de normalização é uma das bases desse modelo. Os termos abaixo são relevantes para o entendimento das três formas normais.

a. CÉLULA

Interseção (LINHA X COLUNA) de uma relação.

b. ITEM REPETITIVO (VALOR NÃO-ATÔMICO ou ATRIBUTO NÃO-SIMPLES).

Ocorre quando uma célula possui mais do que um valor de atributo, é representado por estruturas de dados dos tipos VETOR, MATRIZ ou ITENS DE GRUPO, que impedem a adequada aplicação das operações relacionais, com SQL (Structured Query Language).

c. VALOR ATÔMICO (ATRIBUTO SIMPLES)

Caracterizado quando uma célula possui apenas um valor de atributo. Esta é a situação adequada no modelo Relacional.

d. CHAVE PRIMÁRIA

CHAVE PRIMÁRIA, PRIMARY KEY (PK) ou simplesmente CHAVE é o atributo ou combinação de atributos que permite a IDENTIFICAÇÃO ÚNICA de cada tupla na relação. A PK não admite duplicata e nem valor nulo.

Ex: Se pesquisarmos uma relação de FUNCIONÁRIOS, de PK = MATRICULA, utilizando a matricula como chave de acesso, deveremos obter uma única tupla como resultado da pesquisa.

A chave primária pode ser simples ou composta:

- **SIMPLES:** Constituída de apenas um atributo

Exemplo:

COD-PRODUTO ==> NOME-PROD

NUM-CONTA ==> NOME-CLI, DT-NASC, SALDO

- **COMPOSTA:** formada pela concatenação de dois ou mais atributos.

Exemplo:

COD-PROD + COD-FORNECEDOR ==> PREÇO-PROD

MATR-ALUNO + MATR-PROF + DATA-PROVA ==> NOTA-PROVA

NUM-CONTA + TIPO-APLICAÇÃO + DATA ==> SALDO-APLIC

e. DEPENDÊNCIA FUNCIONAL

É a correspondência (identificação unívoca) existente entre dois atributos de uma mesma relação. pode ser de três tipos: **COMPLETA**, **PARCIAL** e **TRANSITIVA**

f. DEPENDÊNCIA FUNCIONAL COMPLETA (DFC)

Relação de identificação unívoca entre o **ATRIBUTO-CHAVE** e os demais atributos da relação.

Ex: COD-CLIENTE ==> NOME-CLIENTE, ENDEREÇO;

COD-CLIENTE + NUM-PRESTAÇÃO ==> DT-VENCIMENTO, VALOR;

g. DEPENDÊNCIA FUNCIONAL PARCIAL (DFP)

Relação de identificação unívoca entre parte da CHAVE PRIMÁRIA (PK composta por dois ou mais atributos) e algum dos demais atributos da relação.

Ex: COD-PRODUTO + COD-FORNECEDOR ==> NOME-PROD, PREÇO

COD-PRODUTO identifica univocamente o NOME-PROD e é um componente da chave primária.

Obs.: Para que ocorra dependência parcial é necessário chave primária composta. Por outro lado, nem sempre que ocorre PK composta haverá dependência parcial.

h. DEPENDÊNCIA FUNCIONAL TRANSITIVA (DFT)

Relação de identificação unívoca entre atributos que não fazem parte da chave primária da relação.

Ex: PK-MATR ==> NOME, DT-NASC, COD-SETOR, NOME-SETOR

COD-SETOR identifica univocamente o NOME-SETOR e não faz parte da chave.

3. NOTAÇÃO PARA AS ESTRUTURAS DE DADOS

Existem diversas notações, segundo as quais, podemos representar genericamente uma relação. Neste trabalho iremos adotar, principalmente, a notação empregada por CHRIS GANE para a descrição de depósitos de dados e, opcionalmente, a notação de YORDON/DE MARCO.

TABELA VENDA:

				ITENS-DE-VENDA		
NUM-NF	NOME-CLI	END-CLI	DT-VENDA	COD-PROD	QUANT	P-U
				01	10	20
001	Antônio	SQS	22/08	02	20	10
				05	8	5
02	Juliana	SQN	10/09	01	6	20
03	Cláudia	SQS	20/07	05	10	5
.
.
.

A representação genérica da relação VENDA, conforme a notação de GANE, corresponde à seguinte:

VENDA ==> nome da relação

NUM-NF

NOME-CLI

END-CLI

DT-VENDA

ITENS-DE-VENDA*=====> grupo repetitivo

COD-PROD

QUANT

P-UNIT

Observações:

- ITENS DE GRUPO são IDENTADOS, com deslocamento para a direita dando idéia de hierarquia;
- GRUPOS REPETITIVOS são sinalizados com “*” e/ou grafados no PLURAL.
- Os atributos componentes da CHAVE devem receber uma das seguintes notações:
 - . sublinhados, ou;
 - . Um “#” ou um “C” colocados à esquerda dos atributos.

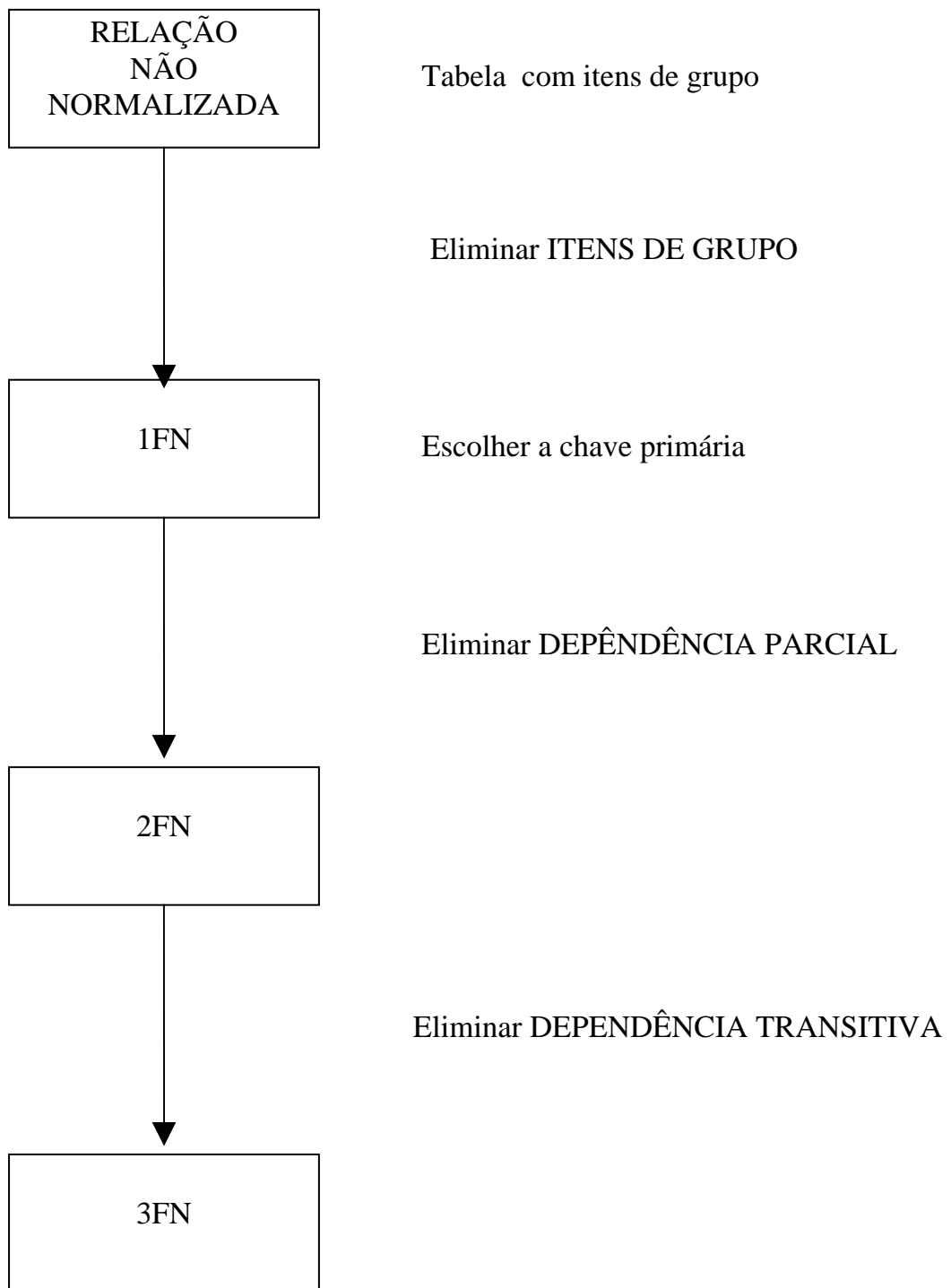
A representação genérica da tabela “VENDA” segundo a notação de YORDON/DE MARCO é:

VENDA = NUM-NF, NOME-CLI, END-CLI, DATA-VENDA, ITENS-DE-VENDA
{COD-PROD, QUANT, P-UNIT}

Observações:

- GRUPOS REPETITIVOS são representados entre chaves;
- O ATRIBUTO-CHAVE deve ser sublinhado.
- Para relações com grande número de atributos a notação de GANE é mais eficiente;

4. ESQUEMA DA NORMALIZAÇÃO



5. RELAÇÕES NÃO-NORMALIZADAS

Uma relação NÃO NORMALIZADA é aquela que possui atributos do tipo NÃO-SIMPLES (NÃO-ATÔMICOS).

Para a devida utilização dos OPERADORES RELACIONAIS é necessário que a relação não-normalizada seja transformada numa forma onde os atributos só contenham VALORES ATÔMICOS, em outras palavras, é preciso tornar a estrutura de dados plana. Esse processo de planificação da relação é concretizado após a sua transposição para a 1FN.

Considere a relação abaixo:

Relação: CONTA CORRENTE

CONTA-CORRENTE

CONTA

AGENCIA

NUMERO

NOME-CLIENTE

ENDEREÇO-CLIENTE

DEPENDENCIA

TIPO-AGENCIA

DESCRIÇÃO-TIPO-AGENCIA

ENDEREÇO-DEPENDENCIA

LANÇAMENTOS*

NUM-DOCUMENTO

DATA-DOCUMENTO

VALOR-LANÇAMENTO Observações:

- Os atributos “CONTA”, “DEPENDÊNCIA” e “LANÇAMENTOS” são itens de grupo;
- O atributo “LANÇAMENTOS” é um grupo repetitivo;
- Esses atributos são do tipo não-atômicos, pois suas células não contém valores únicos.
- A relação “CONTA-CORRENTE” está na forma NÃO-NORMALIZADA.

6. PRIMEIRA FORMA NORMAL (1FN)

Uma relação está em 1FN se todos seus ATRIBUTOS são SIMPLES (ATÔMICOS).

Para colocarmos uma relação em 1FN devemos PLANIFICA-LA, eliminando de sua estrutura os atributos NÃO-ATÔMICOS (VETOR, MATRIZ e ITEM DE GRUPO), de modo que, cada célula da tabela possua apenas um valor de atributo. Isto porque os atributos NÃO-ATÔMICOS não podem ser implementados nos SGBD RELACIONAIS.

A especificação abaixo, corresponde à relação CONTA-CORRENTE após o processo de normalização (1FN):

CONTA-CORRENTE
AGENCIA
NUMERO-CONTA
NOME-CLIENTE
ENDEREÇO-CLIENTE
TIPO-AGENCIA
DESCRIÇÃO-TIPO-AGENCIA
ENDEREÇO-DEPENDENCIA
NUM-DOCUMENTO
DATA-DOCUMENTO
VALOR-LANÇAMENTO

Observações:

- O esquema genérico passou a contar somente com ATRIBUTOS SIMPLES. Todos os ITENS DE GRUPO foram eliminados.
- Assim como toda a relação em 1FN, a estrutura de dados acima apresenta redundâncias e anomalias de atualização.
- CODD estabelece um outro procedimento para normalização (1FN), que é o de decompor a relação não-normalizada em tantas relações quantos forem os grupos repetitivos além de incluir uma relação para o conjunto de colunas atômicas. No processo que descrevemos essas relações surgem naturalmente na derivação das formas normais seguintes (2FN e 3FN).

7. ESCOLHA DA CHAVE PRIMÁRIA

Estando a relação em 1FN, o próximo passo no esquema de normalização é a escolha da CHAVE PRIMÁRIA.

CHAVE PRIMÁRIA é Atributo (chave simples) ou combinação de atributos (chave composta) que identifica univocamente as tuplas de uma relação.

Na escolha do ATRIBUTO-CHAVE os seguintes aspectos são relevantes:

- a. Não pode conter valor nulo para evitar duplicatas;
- b. Não pode conter duplicatas para garantir a identificação unívoca;
- c. Deve ser um atributo estável (não sujeito à constantes mudanças);

Estável: MATRICULA, CPF, NUM-CONTA-CORRENTE

Não estável: MOEDA-NACIONAL, SALDO, INDICE-ECONÔMICO

- d. Não deve dar margem à ambiguidades para garantir a eficiência de acesso (dar preferência a códigos numéricos e o mais curtos possíveis);

Obs1: atributos alfabéticos podem gerar dúvidas quanto à grafia.

Ex: Nome de pessoa - Luís ou Luiz; Melo ou Mello

Nome de órgão - GERAD; GEDAD; GEPAD;

Obs2: Códigos alfanuméricos ou atributos muito extensos são mais propensos a erros de digitação.

- e. Os grupos repetitivos, constantes da relação não-normalizada, devem ceder pelo menos um atributo para formar a chave composta da relação em 1FN;
- f. CHAVES CANDIDATAS ocorrem quando numa relação existem vários atributos (ou combinações) com potencial de CHAVE PRIMÁRIA. Nesse caso, para escolher-se a CHAVE da relação, deve-se considerar os critérios anteriormente definidos. Somente uma CHAVE PRIMÁRIA será escolhida, as demais serão chamadas CHAVES ALTERNATIVAS.

- g. O processo de escolha de CHAVES PRIMÁRIAS em um BD relacional constitui um fator crítico, que afeta a estabilidade do Banco de dados, pois, os relacionamentos são implementados através da redundância das CHAVES. Portanto, qualquer alteração na chave repercute em todos os relacionamentos nos quais a entidade detentora da mesma esteja envolvida (direta ou indiretamente).

Exemplo: Consideremos a relação CONTA-CORRENTE em 1FN (ITEM 5.5):

CONTA-CORRENTE
AGENCIA
NUMERO-CONTA
NOME-CLIENTE
ENDEREÇO-CLIENTE
TIPO-AGENCIA
DESCRIÇÃO-TIPO-AGENCIA
ENDEREÇO-DEPENDENCIA
NUM-DOCUMENTO
DATA-DOCUMENTO
VALOR-LANÇAMENTO

Qual o atributo ou combinação de atributos que identificam singularmente cada tupla da relação CONTA-CORRENTE?

- R1: O atributo “AGÊNCIA-CONTA” isoladamente deve ser descartado, pois, o código de uma agência relaciona-se com diversos números de conta;
- R2: O “NUMERO-CONTA” isoladamente não é adequado, haja vista, que podem existir duas contas com o mesmo número em agências diferentes;
- R3: A combinação “AGÊNCIA + NUMERO-CONTA” ainda não é satisfatória, porque podem existir diversos lançamentos (NUM-DOC, DATA, VALOR) para cada conta vinculada a uma agência;
- R4: Como “LANÇAMENTOS” é um grupo repetitivo na forma NÃO-NORMALIZADA da relação CONTA-CORRENTE, naturalmente, ele deve ceder um atributo para compor a chave primária. Assim, a CHAVE dessa relação é COMPOSTA pela concatenação dos atributos:
- AGÊNCIA + NUMERO-CONTA + NUM-DOC
- R5: Se considerássemos possível dois documentos, com o mesmo número, em sua mesma conta, deveríamos buscar um outro arranjo para a chave-primária.

8. SEGUNDA FORMA NORMAL (2FN)

Uma relação está em 2FN se:

- está em 1FN;

- não contém atributos que dependam funcionalmente de subconjuntos da CHAVE PRIMÁRIA COMPOSTA, em outras palavras, não contém DEPENDÊNCIA FUNCIONAL PARCIAL (DFP).

Para passarmos uma relação da 1FN para a 2FN devemos ELIMINAR as DEPENDÊNCIAS PARCIAIS. Para tanto, utilizamos o conceito de PROJEÇÃO, gerando novas tabelas contendo as colunas que se encontram em DFP com a chave primária. A aplicação da 2FN sobre a relação “CONTA-CORRENTE” resulta na criação das seguintes tabelas:

CONTA

NUMERO-CONTA
NOME-CLIENTE
ENDEREÇO-CLIENTE

AGENCIA

NUM-AGENCIA
TIPO-AGENCIA
DESCRIÇÃO-TIPO-AGENCIA
ENDEREÇO-DEPENDENCIA

LANÇAMENTOS

AGENCIA
NUMERO-CONTA
NUM-DOCUMENTO
DATA-DOCUMENTO
VALOR-LANÇAMENTO

9. TERCEIRA FORMA NORMAL (3FN)

Uma relação está em 3FN se:

- Está em 1FN;
- Está em 2FN;
- Não possui DEPENDÊNCIA FUNCIONAL TRANSITIVA (DFT).

Para passarmos uma relação da 2FN para a 3FN devemos ELIMINAR as DEPENDÊNCIAS TRANSITIVAS utilizando a operação de PROJEÇÃO. Assim, são geradas novas tabelas correspondentes às DFT identificadas. Ao decompor-mos a tabela “CONTA-CORRENTE”, gerando as relações em 2FN, restou apenas uma DFT, que encontra-se na relação “DEPENDÊNCIA”. Fazendo a PROJEÇÃO dessa relação para eliminar a DFT obtemos as relações abaixo:

AGENCIA

NUM-AGENCIA
TIPO-AGENCIA
ENDEREÇO-DEPENDENCIA

TIPO-AGENCIA

TIPO-AGENCIA
DESCRIÇÃO-TIPO-AGENCIA

CONTA

NUMERO-CONTA
NOME-CLIENTE
ENDEREÇO-CLIENTE

LANÇAMENTOS

AGENCIA
NUMERO-CONTA
NUM-DOCUMENTO
DATA-DOCUMENTO
VALOR-LANÇAMENTO

Observações:

- A chave da relação “TIPO-AGÊNCIA” permaneceu na relação principal como CHAVE ESTRANGEIRA, possibilitando o relacionamento entre as duas tabelas.
- As relações “CONTA” e “LANÇAMENTO” já se encontram em 3FN, porque não contém DFT.
- Com a aplicação da 3FN, TODAS as DEPENDÊNCIAS FUNCIONAIS restantes nas relações são do tipo COMPLETAS.

BIBLIOGRAFIA

1. CHU, SHAO YONG
BANCO DE DADOS - ATLAS
2. KORTH, HENRY F.
SISTEMAS DE BANCO DE DADOS - MAC GRAW
3. DATE , C. J.
BANCO DE DAODS - TÓPICOS AVANÇADOS - CAMPUS
4. SETZER, VALDEMAR W.
BANCO DE DADOS
5. ACÁCIO FELICIANO NETO
ENGENHARIA DA INFORMAÇÃO - MAC GRAW
6. GANE, CHRIS
ANÁLISE ESTRUTURADA DE SISTEMAS - LTC
7. GANE, CHRIS
DESENVOLVIMENTO RÁPIDO DE SISTEMAS - LTC
8. YORDON, EDWARD
ANÁLISE ESTRUTURADA MODERNA - CAMPUS
9. CHEN, PETER
MODELO ENTIDADE x RELACIONAMENTOS