Módulo 9

Banco de Dados

Lição 4

Projeto Físico do Banco de Dados

Versão 1.0 - Fev/2009

Autor
Ma. Rowena C. Solamo

Equipe
Rommel Feria
Rick Hillegas
John Paul Petines

Necessidades para os Exercícios

Sistemas Operacionais Suportados

NetBeans IDE 5.5 para os seguintes sistemas operacionais:

- Microsoft Windows XP Profissional SP2 ou superior
- Mac OS X 10.4.5 ou superior
- Red Hat Fedora Core 3
- Solaris™ 10 Operating System (SPARC® e x86/x64 Platform Edition)

NetBeans Enterprise Pack, poderá ser executado nas seguintes plataformas:

- Microsoft Windows 2000 Profissional SP4
- Solaris™ 8 OS (SPARC e x86/x64 Platform Edition) e Solaris 9 OS (SPARC e x86/x64 Platform Edition)
- Várias outras distribuições Linux

Configuração Mínima de Hardware

Nota: IDE NetBeans com resolução de tela em 1024x768 pixel

Sistema Operacional	Processador	Memória	HD Livre
Microsoft Windows	500 MHz Intel Pentium III workstation ou equivalente	512 MB	850 MB
Linux	500 MHz Intel Pentium III workstation ou equivalente	512 MB	450 MB
Solaris OS (SPARC)	UltraSPARC II 450 MHz	512 MB	450 MB
Solaris OS (x86/x64 Platform Edition)	AMD Opteron 100 Série 1.8 GHz	512 MB	450 MB
Mac OS X	PowerPC G4	512 MB	450 MB

Configuração Recomendada de Hardware

Sistema Operacional	Processador	Memória	HD Livre
Microsoft Windows	1.4 GHz Intel Pentium III workstation ou equivalente	1 GB	1 GB
Linux	1.4 GHz Intel Pentium III workstation ou equivalente	1 GB	850 MB
Solaris OS (SPARC)	UltraSPARC IIIi 1 GHz	1 GB	850 MB
Solaris OS (x86/x64 Platform Edition)	AMD Opteron 100 Series 1.8 GHz	1 GB	850 MB
Mac OS X	PowerPC G5	1 GB	850 MB

Requerimentos de Software

NetBeans Enterprise Pack 5.5 executando sobre Java 2 Platform Standard Edition Development Kit 5.0 ou superior (JDK 5.0, versão 1.5.0_01 ou superior), contemplando a Java Runtime Environment, ferramentas de desenvolvimento para compilar, depurar, e executar aplicações escritas em linguagem Java. Sun Java System Application Server Platform Edition 9.

- Para **Solaris**, **Windows**, e **Linux**, os arquivos da JDK podem ser obtidos para sua plataforma em http://java.sun.com/j2se/1.5.0/download.html
- Para **Mac OS X**, Java 2 Plataform Standard Edition (J2SE) 5.0 Release 4, pode ser obtida diretamente da Apple's Developer Connection, no endereço: http://developer.apple.com/java (é necessário registrar o download da JDK).

Para mais informações: http://www.netbeans.org/community/releases/60/relnotes.htm

Java™ DB System Requirements

Java™ DB is supported on the Solaris, Linux and Windows operating systems and Sun Java 1.4 or later.

Colaboradores que auxiliaram no processo de tradução e revisão

Aécio Júnior
Alberto Ivo da Costa Vieira
Alexandre Mori
Alexis da Rocha Silva
Aline Sabbatini da Silva Alves
Allan Wojcik da Silva
Angelo de Oliveira
Aurélio Soares Neto
Bruno da Silva Bonfim
Carlos Fernando Gonçalves
Carlos Hilner Ferreira Costa
Daniel Noto Paiva
Daniel Wildt
Denis Mitsuo Nakasaki
Fábio Antonio Ferreira
Givailson de Souza Neves
Jacqueline Susann Barbosa
Jader de Carvalho Belarmino
João Vianney Barrozo Costa
José Francisco Baronio da Costa
Kleberth Bezerra Galvão dos Santos
Luiz Fernandes de Oliveira Junior
Maria Carolina Ferreira da Silva
Maricy Caregnato
Mauricio da Silva Marinho
Paulo Oliveira Sampaio Reis
Ronie Dotzlaw
Seire Pareja
Sergio Terzella
Thiago Magela Rodrigues Dias

Auxiliadores especiais

Revisão Geral do texto para os seguintes Países:

- **Brasil** – Tiago Flach
- **Guiné Bissau** – Alfredo Cá, Bunene Sisse e Buon Olossato Quebi – ONG Asas de Socorro

Coordenação do DFJUG

- **Daniel deOliveira** – JUGLeader responsável pelos acordos de parcerias
- **Luci Campos -** Idealizadora do DFJUG responsável pelo apoio social
- **Fernando Anselmo -** Coordenador responsável pelo processo de tradução e revisão, disponibilização dos materiais e inserção de novos módulos
- **Rodrigo Nunes -** Coordenador responsável pela parte multimídia
- **Sérgio Gomes Veloso -** Coordenador responsável pelo ambiente JEDI™ (Moodle)

Agradecimento Especial

John Paul Petines – Criador da Iniciativa JEDI™
Rommel Feria – Criador da Iniciativa JEDI™

1. Objetivos

Nesta lição, discutiremos o projeto físico do banco de dados para bancos de dados relacionais. Nesse capítulo, iremos discutir o modelo físico da base de dados a partir da base de dados lógica. Regras de negócio e restrições de integridade que inclui o tópico de restrições de domínio, integridade da entidade, integridade referencial e operações de gatilho. Quando estamos executando a análise de volume de dados, se estima o tamanho da base de dados que é usado para selecionar os dispositivos de armazenamento físico e estimar o custo de armazenamento.

Realizarmos uma análise de uso e estimar os caminhos ou padrões usados para selecionar a organização de arquivos e métodos de acesso para planejar o uso de índices e uma estratégia de distribuição dos dados.

A memória principal dos computadores é inadequada para o armazenamento da base de dados. Dispositivos secundários de armazenagem como discos rígidos e discos ópticos disponibilizam um meio para armazenar fisicamente a base de dados. A organização de arquivos e métodos de acesso são discutidos para entender como a base de dados são armazenadas fisicamente no armazenamento secundário. Para ajudar na eficiência do retorno de linhas na base de dados, índices foram discutidos, incluindo os diferentes tipos: nomeados, índices seqüenciais, não-seqüenciais, índice em cluster, índices em multi-níveis e árvores B/B+. Mostrar o conceito de desnormalização para otimizar certos processamentos de base de dados.

Ao final desta lição, o estudante será capaz de:

- Realizar o projeto físico do banco dados
- Conhecer as regras de negócio e restrições de integridade
- Formular o volume de dados e análise de uso
- Organizar os arquivos e conhecer os métodos de acesso
- Saber mais a respeito dos índices

2. Projeto Físico do Banco de Dados

É o processo de mapear as estruturas lógicas do banco de dados desenvolvidas nos estágios anteriores em um modelo interno ou um conjunto de estruturas físicas do banco de dados. Deve ser feito com cuidado já que as decisões tomadas neste estágio têm maior impacto na acessibilidade dos dados, tempo de resposta, segurança, uso amigável e fatores similares. O maior objetivo é implementar o banco de dados como um conjunto de registros armazenados, arquivos, índices e outras estruturas de dados que irão oferecer a performance adequada e assegurar a integridade do banco de dados, segurança e recuperação.

A fase anterior ao projeto físico, especificamente o projeto lógico do banco de dados, é altamente independente dos detalhes de implementação; sem funcionalidades específicas dos SGBD's e programas de aplicação alvos.

A saída do projeto lógico do banco de dados, por exemplo, o esquema relacional e o dicionário de dados são as fontes para o processo de projeto físico. Eles oferecem ao projetista de banco de dados uma maneira de fazer negociações que são muito importantes para um projeto de banco de dados eficiente.

O projeto lógico do banco de dados está relacionado com o *quê*; projeto físico do banco de dados está relacionado com o *como*. Ele requer habilidades que são geralmente encontradas em pessoas diferentes. Ele deve saber como o sistema de computador onde reside o SGBD opera e deve estar completamente consciente da funcionalidade do SGBD alvo.

As Três Maiores Entradas para o Projeto Físico do Banco de Dados:

1. Estrutura lógica do banco de dados (Esquema Relacional)
2. Requisitos de processamento do usuário que foram identificados durante a definição dos requisitos, incluindo o tamanho e a freqüência de utilização do banco de dados.
3. Características do sistema de gerenciamento do banco de dados (SGBD) e outros componentes do ambiente de operação do computador.

O primeiro passo para transformar um projeto lógico de banco de dados em seu projeto físico de banco de dados envolve a tradução das relações derivadas do processo do projeto lógico do banco de dados em uma forma que possa ser implementada no SGBD relacional alvo. Isto requer conhecimento das funcionalidades oferecidas pelo SGBD alvo tais como:

- Se o sistema suporta a definição de chaves primárias, chaves estrangeiras e chaves alternativas
- Se o sistema suporta a definição de dados obrigatórios
- Se o sistema suporta a definição de domínios
- Se o sistema suporta a definição de restrições de negócio
- Como criar as relações base

Os objetivos de se definir a especificação no SGBD alvo é produzir um esquema básico de trabalho do banco de dados relacional das estruturas do projeto lógico do banco de dados que envolve decisões em como representar as relações base e projetar as restrições de negócio para o SGBD alvo. Para especificar o projeto físico das tabelas, precisamos considerar o seguinte:

- Regras de Negócio e Restrições de Integridade
- Volume de Dados e Análise de Uso
- Estratégias de Distribuição de Dados
- Organização do Arquivo e Métodos de Acesso ao Arquivo
- Índices
- Normalização

3. Regras de Negócio ou Restrições de Integridade

Regras de Negócio ou ***Restrições de Integridade*** são especificações que preservam a integridade do modelo lógico de dados. O termo *regras de negócio* é geralmente usado no contexto da fase de análise do banco de dados enquanto o termo *restrições de integridade* é usado no contexto da fazer de projeto. Na fase de análise, o modelo conceitual ou Modelo ER está primariamente interessado na estrutura dos dados ao invés de expressar as regras de negócio. Entretanto, se uma regra de negócio é descoberta, ela deve ser documentada. Nas próximas seções do capítulo, o termo restrições de integridade será usado.

Na maioria dos sistemas de gerenciamento de banco de dados, eles incorporam as restrições de integridade dentro do escopo do sistema de banco de dados em vez de dentro dos programas de aplicação ou operadores humanos. Algumas das vantagens de se colocar as restrições de integridade nos sistema de gerenciamento de banco de dados são:

1. Oferece desenvolvimento mais rápido de aplicações com poucos erros.
2. Reduz o esforço e o custo de manutenções.
3. Oferece resposta mais rápida às mudanças de negócio.
4. Facilita o envolvimento do usuário final no desenvolvimento de novos sistemas e na manipulação dos dados.
5. Oferece aplicações consistentes das restrições de integridade.
6. Reduz o tempo e esforço necessário para treinar programadores de aplicação.
7. Promove facilidade de utilização de um banco de dados.

Três Categorias de Restrições de Integridade:

1. Restrições de Domínio
2. Integridade da Entidade
3. Integridade Referencial
4. Operações de Gatilhos (*Triggers*)

A entrada básica pra o processo de projeto físico é a relação ou tabela. Como revisão, uma ***tabela*** é um representação bidimensional de dados ou entidades consistindo de uma ou mais colunas, e zero ou mais linhas. Figura 1 mostra um exemplo de uma relação ou tabela com os seus componentes correspondentes. Esta é a tabela ou relação STORE.

Nome da Tabela ----------->> STORE

Atributos ----------->>

Amostra de Valores ----------->>

Number	Name	Address
10	GangStore in Alabama	Alabama
20	GangStore in West Virginia	West Virgini
30	GangStore in North Dakota	NorthDakota

Figura 1: Tabela ou Relação Store

Na nomeação de tabelas e colunas, as seguintes regras devem ser observadas:

- Os nomes das tabelas devem ser únicos. Dentro do modelo de dados, não deve haver duas tabelas com o mesmo nome.
- Os nomes das colunas devem ser únicos dentro de uma tabela. Entretanto, colunas em tabelas diferentes podem compartilhar o mesmo nome.
- As linhas devem ser únicas. As linhas consideradas em sua totalidade devem ser distintas umas das outras.

3.1. Restrições de Domínio

Um entendimento de domínios é importante na definição de restrições de integridade. Um ***domínio*** é um conjunto de todos os tipos de dados e série de valores que atributos podem assumir. Tipicamente, especifica:

- significado
- tipo do dado
- valores permitidos
  - unicidade
  - suporte a nulo
  - intervalo
- formato
  - tamanho
  - projeto do código
- série

As vantagens do uso de domínios são:

1. Valida os valores de um atributo.
2. Ajuda a poupar esforço na descrição das características do atributo.

Exemplos de restrições de domínio são dadas na Figura 2. Um ACCOUNT tem alguns WITHDRAWALs.

ACCOUNT

ACCOUNT NO.	...	BALANCE
SAR-1034	...	12987,34
SAR-1125	...	3000,00
CAR-1256	...	5126,14
CAR-3756	...	9834,33

WITHDRAWAL

ACCOUNT NO.	TRANSACTION DATE	...	AMOUNT
SAR-1034	25/03/01 15:00	...	500,00
SAR-1034	01/04/01 11:30	...	1500,00
CAR-3756	22/01/01 09:30	...	3000,00

```
Atributo: ACCOUNT NO.
Significado: Número da conta do cliente no banco
Tipo do Dado: Caracter
Formato: AAA-AAAA
Unicidade: Deve ser único
Suporte a Nulo: Não-nulo

Atributo: AMOUNT
Significado: Quantidade do Saque
Tipo do Dado: 2 Casas Decimais
Intervalo: 0-P10,000
Unicidade: Não-único
Suporte a Nulo: Não-nulo
```

Figura 2: Exemplos de Restrições de Domínio

Tipos de Dados

Para cada coluna identificada, define os tipos de dados válidos. Podemos usar tipos de dados de diferentes linguagens de programação para representar os tipos de dados válidos que uma coluna pode ter. Neste curso, usaremos os tipos de dados JavaDB que são compatíveis com o SQL.

Tipos de Dados Numéricos

Tipos Numéricos incluem os seguintes tipos, que oferecem armazenamento de tamanhos variados:

- Numéricos Inteiros
  - SMALLINT
    - Sintaxe: SMALLINT
    - Oferece 2 bytes de armazenamento.
    - O intervalo é de -32.768 a 32.767.
  - INTEGER
    - Sintaxe: {INTEGER | INT}
    - Oferece 4 bytes de armazenamento para valores inteiros.
    - O intervalo é de -2147483648 a 2147483647.
  - BIGINT
    - Sintaxe: BIGINT
    - Oferece 8 bytes de armazenamento para valores inteiros.
    - O intervalo é de -9223372036854775808 a 9223372036854775807.
- Numéricos Aproximados ou de Ponto-Flutuante
  - REAL
    - Sintaxe: REAL

- Oferece 4 bytes de armazenamento para números usando a notação de ponto-flutuante IEEE.
- Constantes numéricas de ponto-flutuante são limitadas a 30 caracteres de comprimento.
- Limitação dos seus intervalos:
  - Menor valor REAL: -3.402E+38
  - Maior valor REAL: 3.402E+38
  - Menor valor positivo REAL: 1.175E-37
  - Maior valor negativo REAL: -1.175E-37

- DOUBLE PRECISION
  - Sintaxe: {DOUBLE PRECISION | DOUBLE}
  - Oferece 8 bytes de armazenamento para números usando a notação de ponto-flutuante IEEE.
  - É o sinônimo de DOUBLE.
  - Constantes numéricas de ponto-flutuante são limitadas a 30 caracteres de comprimento.
  - Limitação dos seus intervalos:
    - Menor valor DOUBLE: -1.79769E+308
    - Maior valor DOUBLE: 1.79769+308
    - Menor valor positivo DOUBLE: 2.225E-307
    - Maior valor negativo DOUBLE: -2.225E-307

- Número Exato
  - DECIMAL (armazenamento baseado na precisão)
    - Sintaxe: {DECIMAL | DEC}[(precisão [,escala])]
    - Oferece um numérico exato no qual a precisão e a escala podem ser atribuídos arbitrariamente.
    - A quantidade de armazenamento necessário será baseado na precisão.
    - A precisão definirá o número total de dígitos, tanto para a esquerda quanto para a direita do ponto decimal; a escala definirá o número de dígitos do componente fracionário.
    - A precisão deve estar entre 1 e 31; a escala deve ser menor ou igual à precisão.

Tipos de Dados Caracteres

Tipos Caracteres incluem os seguintes:

- Tipos Caractere de Tamanho-fixo
  - CHAR
    - Sintaxe: CHAR[ACTER][(tamanho)]
    - Oferece armazenamento de strings de tamanho fixo.
    - O tamanho é um inteiro constante sem sinal. O tamanho padrão é 1.
    - Espaços completam um valor de string menor que o tamanho esperado. Espaços remanescentes são truncados se você tentar inserir um string de tamanho superior. Ocorre erro se o string resultante terminar sem espaços e ainda assim for muito comprido.

- CHAR FOR BIT DATA
  - Sintaxe: {CHAR | CHARACTER}[(tamanho)] FOR BIT DATA
  - Permite armazenar strings de bytes para um tamanho específico.
  - É útil para dados não estruturados onde strings de caracteres não são apropriados.
  - O tamanho é um literal inteiro sem sinal designando o tamanho em bytes.
  - O tamanho padrão é 1; o tamanho máximo é 254 bytes.

- Tipos Caractere de Tamanho-variável
  - VARCHAR
    - Sintaxe: {VARCHAR | CHAR VARYING | CHARACTER VARYING} (tamanho)
    - Oferece armazenamento para strings de tamanho variável.
    - O tamanho é um inteiro constante sem sinal, e não deve ser maior que a restrição do inteiro usado para especificar o tamanho.
    - O tamanho máximo é 32.672 caracteres.
    - Espaços não são incluídos se o valor é menor que o tamanho.
    - Espaços finais são truncados para o tamanho da coluna. Se o string resultante continuar grande, um erro ocorre.
  - LONG VARCHAR
    - Sintaxe: LONG VARCHAR
    - Permite armazenamento de caracteres de um tamanho máximo de 32.700 caracteres.
    - É similar ao VARCHAR exceto que o número máximo de caracteres não é especificado.
  - VARCHAR FOR BIT DATA
    - Sintaxe: {VARCHAR | CHAR VARYING | CHARACTER VARYING} (tamanho) FOR BIT DATA
    - Permite armazenar strings binários menores ou iguais ao tamanho especificado.
    - É útil para dados não estruturados onde strings de caracteres não são apropriados (tais como nas imagens).
    - O tamanho é um inteiro constante sem sinal definindo o tamanho em bytes.
    - Não tem tamanho padrão ao contrário do CHAR FOR BIT DATA.
    - O tamanho máximo do tamanho é 32.672 bytes.
  - LONG VARCHAR BIT DATA
    - Sintaxe: LONG VARCHAR FOR BIT DATA
    - Permite o armazenamento de strings de bits até 32.700 bytes.
    - É idêntico ao VARCHAR FOR BIT DATA exceto que o tamanho máximo não está especificado.

Tipos de Dados de Data e Hora

- DATE
  - Sintaxe: DATE
  - Oferece armazenamento de um ano-mês-dia em um intervalo suportado pelo java.sql.Date.

- Suporta os seguintes formatos:
  - aaaa-mm-dd
  - mm/dd/aaaa
  - dd.mm.aaaa
- Não deve ser misturado com TIME e TIMESTAMP em uma expressão.

- TIME
  - Sintaxe: TIME
  - Oferece armazenamento de um valor hora-do-dia.
  - Suporta os seguintes formatos:
    - hh:mm[:ss]
    - hh.mm[.ss]
    - hh[:mm] {AM | PM}
- TIMESTAMP
  - Sintaxe: TIMESTAMP
  - Armazena um valor combinado de DATE e TIME.
  - Permite fracionamento de segundos até nove dígitos.
  - Suporta os seguintes formatos:
    - aaaa-mm-dd hh:mm:ss[.nnnn]
    - aaaa-mm-dd-hh.mm.ss[.nnnn]

Tipos de Dados para Grandes Objetos

- CLOB
  - Sintaxe: {CLOB | CHARACTER LARGE OBJECT} [(tamanho [{K | M| G}])]
  - É conhecido como objeto de caractere grande com um valor até 2.147.483.647 de comprimento.
  - É usado para armazenar dados baseados nos caracteres unicode, tais como grandes documentos em qualquer conjunto de caracteres.
  - O tamanho é dado em número de caracteres em ambos CLOB, a menos que um dos sufixos K, M ou G é dado, referindo-se ao múltiplos de 1024, 1024*1024, 1024*1024*1024, respectivamente.
- BLOB
  - Sintaxe: {BLOB | BINARY LARGE OBJECT} [(tamanho [{K|M|G}])]
  - É um string binário de tamanho variável que pode ser de até 2.147.483.647 caracteres de comprimento.
  - Assim como outros tipos binários, não está associado a uma página de código; não armazena dados caracteres.
  - O tamanho é dado em bytes a menos que um dos sufixos, K, M or G, é dados, referindo-se aos múltiplos de 1024, 1024*1024, 1024*1024*1024, respectivamente.

Outros Tipos de Dados

- XML
  - Sintaxe: XML

- É usado para documentos Extensible Markup Language (XML).
- É usado:
  - para armazenar documentos XML que estão de acordo com a definição SQL/XML de um bem formado valor XML(DOCUMENT(ANY)).
  - Provisoriamente para valores XML(SEQUENCE), que podem não ser um valor bem formado XML(DOCUMENT(ANY)).

STORE

Number	Name	Address
SMALLINT	VARCHAR(250)	VARCHAR(250)
10	GangStore in Alabama	Alabama
20	GangStore in West Virginia	West Virginia
30	GangStore in North Dakota	NorthDakota

EMPLOYEE

Number	LastName	FirstName	MiddleName	Store	Manager	Date_Hired
SMALLINT	VARCHAR(100)	VARCHAR(100)	CHAR(4)	SMALLINT	SMALLINT	DATE
5001	Cruz	Juan	M.	10	5000	25 de Dez de 2005
5002	Enriquez	Sheila	S.P.	10	5001	4 de Jun de 2008
5003	Choo	James	Y.	10	5000	16 de Jan de 1997
5004	Mendes	Lani	M.	10	5001	4 de Jan de 1998

Figura 3: Especificando Tipos de Dados

Valores Permitidos

O que deve ser considerado a seguir são os valores permitidos para cada coluna. Algumas considerações são os valores nulos, valores duplicados, valores modificáveis, chaves e dados derivados ou calculados.

Valores Nulos

Um valo ***nulo*** é um valor que está faltando ou desconhecido em uma coluna de uma tabela. Eles não são o mesmo que brancos. Dois brancos são considerados iguais em valor; a equivalência de dois valores nulos é indeterminado. Embora, eles possam aparecer na tabela como brancos.

Nulos não são iguais a zeros. A maioria das operações aritméticas podem ser realizadas nos valores zero; nulos devem ser excluídos das manipulações matemáticas.

NN significa nulo não-permitido. Para indicar que nulos não são permitidos na coluna de uma tabela, coloque letras NN diretamente abaixo do devido cabeçalho da coluna.

STORE

Number	Name	Address
NN	**NN**	
10	GangStore in Alabama	Alabama
20	GangStore in West Virginia	West Virginia
30	GangStore in North Dakota	NorthDakota

EMPLOYEE

Number	LastName	FirstName	MiddleName	Store	Manager	Date_Hired
NN	**NN**	**NN**		**NN**		**NN**
5001	Cruz	Juan	Martinez	10	5000	25 de Dez de 2005
5002	Enriquez	Sheila	San Pedro	10	5001	4 de Jun de 2008
5003	Ferrer	Grace	Atienza	20	5000	16 de Jan de 1997
5004	Franz	Jane	Solamo	20	5003	4 de Jan de 1998

Figura 4: Especificando Valores Não-nulos

Figura 4 mostra um exemplo de especificação de valores de não-nulos nas tabelas STORE e EMPLOYEE. As colunas Number e Name da tabela STORE são não-nulas. Similarmente, as colunas Number, LastName, Firstname, Store e Date_Hired da tabela EMPLOYEE são não-nulas.

Valores Duplicados

Um ***valor duplicado*** é um valor em uma coluna de uma tabela que exatamente se iguala a algum outro valor naquela mesma coluna.

- **ND** significa nenhuma duplicação permitida. Enquanto a duplicação de uma linha inteira não é permitida, valores duplicados em uma coluna particular de tabelas são comuns.
- Quando a combinação de valores de certas colunas da tabela precisam ser únicas, podemos usar numeração nas marcas **ND** para representar o grupo de colunas. Para mostrar um exemplo, na tabela inventory, a combinação dos valores das colunas store_no e item_code deve ser única na tabela. **ND1** representa o agrupamento.

INVENTORY

Store_no	Item_code	Quantity	Op_Level
NN,ND1	**NN,ND1**	**NN**	**NN**
10	1001	2345	500
10	1006	3245	100
20	1001	456	100

EMPLOYEE

Number	LastName	FirstName	MiddleName	Store	Manager	Date_Hired
NN,ND	**NN**	**NN**		**NN**		**NN**
5001	Cruz	Juan	Martinez	10	5000	25 de Dez de 2005
5002	Enriquez	Sheila	San Pedro	10	5001	4 de Jun de 2008
5003	Ferrer	Grace	Atienza	20	5000	16 de Jan de 1997
5004	Franz	Jane	Solamo	20	5003	4 de Jan de 1998

Figura 5: Especificando Valores Não-Duplicados

Figura 5 mostra como especificar valores não duplicados. Para a tabela STORE, os valores combinados de Number e Name devem ser únicos. Para a tabela EMPLOYEE, o Number deve ser único.

Valores Alteráveis

Um ***valor alterável*** é um valor em uma tabela que pode variar com o tempo. A maioria dos valores na maioria das tabelas são alteráveis.

- **NC** significa que nenhuma alteração é permitida.

STORE

Number	Name	Address
NN,ND1,NC	**NN,ND1**	
10	GangStore in Alabama	Alabama
20	GangStore in West Virginia	West Virginia
30	GangStore in North Dakota	NorthDakota

EMPLOYEE

Number	LastName	FirstName	MiddleName	Store	Manager	Date_Hired
NN,ND,NC	**NN**	**NN**		**NN**		**NN,NC**
5001	Cruz	Juan	Martinez	10	5000	25 de Dez de 2005
5002	Enriquez	Sheila	San Pedro	10	5001	4 de Jun de 2008
5003	Ferrer	Grace	Atienza	20	5000	16 de Jan de 1997
5004	Franz	Jane	Solamo	20	5003	4 de Jan de 1998

Figura 6: Especificando Valores Não-alteráveis

Figura 6 apresenta um exemplo de especificação de valores não alteráveis. Na a tabela STORE (LOJA), o Número (Number) e o Nome (Name) não são permitidos serem alterados uma vez inseridos na tabela. Na tabela EMPLOYEE (FUNCIONÁRIOS), o Número (Number) e a Data de Admissão (Date_Hired) não são alteráveis.

Chaves

Dois tipos de chaves necessitam serem especificadas. Nominalmente, a chave primária e a chave estrangeira.

A **chave primária** de uma tabela é a coluna ou grupo de colunas que possuem valores únicos identificando cada linha da tabela. As regras devem ser observadas:

- Toda tabela deve ter uma chave primária.
- Cada linha de dados deve ser sempre identificável unicamente.
- Toda tabela deve ter apenas uma chave primária.
- PK significa chave primária; todas as chaves primárias obedecem as seguintes regras:
  - Valores da chave primária nunca devem ser nulos.
  - Valores da chave primária nunca devem ser duplicados.
  - Valores da chave primária nunca devem ser alteráveis.
- A chave primária pode ser determinada pelo sistema. **SA** significa determinada pelo sistema (system-assigned). Se o valor da chave primária é automaticamente gerada pelo sistema quando uma nova linha é adicionada, esses valores são ditos para serem determinados pelo sistema, e são marcados como PK,SA. Isto é verdadeiro até mesmo quando a determinação é feita manualmente mas de uma lista gerada por computador.
- A chave primária pode ser determinada pelo usuário. **UA** significa determinada pelo usuário. Se o valor de uma chave primária é especificada por usuários do sistema, tais valores são ditos como determinados pelo usuário e são marcados como PK,UA.
- Observe que as marcações NN, ND e NC já não são necessárias uma vez que estas restrições já estão incluídas na definição da chave primária.

A chave estrangeira é uma coluna ou grupo de colunas que são chaves primárias de outras tabelas.

- **FK** significa chave estrangeira (foreign key). Chaves estrangeiras de multiplas colunas são numeradas como multiplas colunas ND constraints.

STORE

Number	Name	Address
PK,UA,ND1	**NN,ND1**	
10	GangStore in Alabama	Alabama
20	GangStore in West Virginia	West Virginia
30	GangStore in North Dakota	NorthDakota

EMPLOYEE

Number	LastName	FirstName	MiddleName	Store	Manager	Date_Hired
PK,UA	**NN**	**NN**		**NN,FK**		**NN,NC**
5001	Cruz	Juan	Martinez	10	5000	25 de Dez de 2005
5002	Enriquez	Sheila	San Pedro	10	5001	4 de Jun de 2008
5003	Ferrer	Grace	Atienza	20	5000	16 de Jan de 1997
5004	Franz	Jane	Solamo	20	5003	4 de Jan de 1998

Figura 7: Especificando chaves

Figura 7 apresenta como especificar chaves. Para a chave primária, o NN, ND e NC são removidos e substituídos por PK. Na tabela STORE, a chave primária é Number e é determinada pelo usuário. Na tabela EMPLOYEE, a chave primária é Number e é determinada pelo usuário. A coluna Store é uma chave estrangeira que referencia a chave primária da tabela STORE.

Derivado ou Dado Calculado

Dados derivados são dados que são calculados a partir de dados em um modelo definido em outro local. São dados redundantes que complicam a operação de atualização nas tabelas. Devem ser evitados e incluídos no modelo apenas quando graves problemas de perfomance ditar as suas necessidades.

- **DD** significa dados derivados.

Formatos e Projeto de Código

Códigos são um conjunto de números inteiros ou letras usadas para representar a descrição de um item de forma curta para um processamento eficiente. É usado para:

- Identificação e Recuperação
  - Número da Conta
  - Código Item
- Classificação
  - Tipo Conta
  - Status Conta
  - Tipo Demanda
- Eficiente classificação e indexação
  - Códigos curtos resultam em menor tempo de indexação e classificação
- Economia de custos na preparação dos dados
  - Nº Conta substituindo Nome do Cliente
  - Tipo de Conta ao invés de "Residencial" ou "Comercial"

Orientações para Projeto de Código

Algumas orientações para projetar códigos para os valores das colunas.

- Códigos que são:
  - Únicos
    - devem fornecer um valor de código na tabela
  - Numéricos
    - devem ser chaves fáceis de codificar
    - devem ser limitadas a 7-10 dígitos ou menos
  - Alfabéticos
    - devem ser fáceis de serem lembrados
    - devem ser limitadas de 5 a 7 letras ou menos
- Não misture Numéricos e Alfabéticos; exemplos:
  - X989W34H5 é horrível
  - PEC 490 é permitido
  - Códigos devem ser coerentes no formato
  - Para códigos de 4 dígitos atribua 1000 – 9999
- Não utilize baixos valores não significativos precedidos por zero como
  - 0001 pode ser escrito como 1
- Códigos que são protegidos por um dígito verificador:

- é utilizador para evitar erros
- Utilizado quando alta precisão é necessária

Tipos de Esquema de Códigos

A seguir estão alguns esquemas ou estratégias usadas para projetar o formato dos códigos:

1. Esquemas de Códigos Alfanuméricos
    - Código Alabéticos de Tamanho Fixo
        - Códigos Monetários – Usado pelo Banco Mundial, FMI e IATA

          Formato: XXX
            - 2 Primeiros Caracteres – Código do País
            - Últimos Caracteres – Código da Moeda

        Exemplos:

        PHP Filipinas Peso
        SGD Singapura Dólar
        HKD Hong Kong Dólar
        USD US Dólar
        JPY Japão Yen
    - Códigos Alfabéticos de Tamanho Fixo
        - Unidades de Medida – Usado pela Marinha US e Mattel

          Formato: XX Código de duas letras

        Exemplos:

        MT Tonelada Métrica
        PC Pedaço
        ST Conjunto
        Li Litro
        KM Quilometro
    - Código Alfabético de Tamanho Variável
        - Unidades de Medida – Uso de Abreviações Aceitas Universalmente

          Formato: XXX 1 a 3 códigos de caracteres

        Exemplos:

        SET Conjunto
        PCS Pedaços
        LBS Libras
        L Litros
        KM Quilômetros
        M Metros
    - Derivações Alfabéticas
        - Usadas em programas e abreviações

          Formato: Um a sete caracteres

        Exemplo:

        Andrews ANDRWS
        Smith SMTH
        Counter CTR
        Accumulator AC
        Save Area SA
        Switch SW ou SWC

2. Modo Combinado de Esquema de Codificação
    - Códigos de Som ou Códigos Fonéticos

      Formato: X999

      - Peque a primeira letra
      - Ignore todas as vogais A, E, I, O, U e as letras W, H, Y
      - Troque as próximas três letras por:

        1 B,F,P,V
        2 C,G,J,K,Q,S,X,Z
        3 D,T
        4 L
        5 M,N
        6 R

    Exemplo:

        Diaz, Mario Avanceña
        D 2 5 6
        Uy, Whu Yu
        U 0 0 0
        Chua, Joy Uy
        C 2 0 0

    - Imposto Antigo do Nº da Conta (TAN)

      Formato: X999<u>9</u> – X999 – <u>X</u> – <u>9</u>

      Soundex

      Aniversário

      Sublinhado – Cartões Alegres

      Note: X do Aniversário (I não é usado)

JAN	A	JUL	G
FEV	B	AGO	H
MAR	C	SET	J
ABR	D	OUT	K
MAi	E	NOV	L
JUN	F	DEZ	M

      Exemplos:

    1. Rizal, Joe Concepcion born Mar. 12, 1955
       R242? - C1255 - ? - ?

       pode ser R2428 – C1255 – A – 7

       ou R2420 – C1255 – S – 0

- Codificando Nº da Placa
  Formato: XXX 999
  - Código de 3 letras – determinado pela região
    - N Reservado – veículos públicos
    - S – veículos governamentais
    - O I – não utilizados
  - Series – 001 a 999
    Exemplos:
    - PEC 490 veículo privado em Metro Manila
    - SCR 152 veículo governamental
      - NTX 442 veículo público

3. Esquemas Numéricos de Codificação

- Seqüência
  - Normalmente utilizada quando não há código existente. Os valores a serem codificados são dispostos alfabeticamente e números consecutivos são atribuídos. Os números restantes são utilizados para qualquer novo nome.

100 – Aaron, James
102 – Abalos, Jane
:
465 – Mendoza, Estilito
466 – Mendoza, Fred
:
875 – Zapra, Jose

  - Pré-alocação de um conjunto de números para representar um grupo de produtos ou informação
    - 10000 – 19999

Galvanized Steel Sheet Metal
    - 20000 – 29999

Steel Plates
    - 30000 – 49999

Pre-fabricated Building Support
    - 50000 – 79999

Subassemblies

- Codificação de Dígito Significante
  - Certa porção do código representa alguma característica significante da informação a ser codificada

9999
primeiro dígito Faixa de Voltagem
últimos 3 dígitos Watts

Examples:
    - 2025 Luzes 240v 25w
    - 2040 40w
    - 2100 100w
    - 1050 Luzes 110v 50w
    - 1080 80w

- Código de Aparência
  - Este tipo de código é semelhante ao código bloqueado mas tem mais sub-classificações no âmbito de cada faixa de código

NATO Número Armazenado

9999 99 999 9999

Grupo Classe Nação Item Número

- Exemplo de Código de Aparência
  - Livro Geral de Codificação

9 9999 999

    - 1 – Ativo
    - 2 – Passivo
    - 3 – Acionistas
    - 4-5 – Salário
    - 6-9 – Despesas

1XXX Ativo

1100 Caixa
1110 Caixa em banco
1111 Poupança
1112 Far East Bank

4. Códigos Numéricos Protegidos

- Certos códigos requerem alta precisão na codificação para garantir que os valores estão corretos, um dígito verificador é adicionado ao código em ordem para proteger dos seguintes tipos de erros:

3 9 1 7 5 escrito como...

3 9 7 5 Omissão
3 9 1 7 5 2 Adição
8 9 1 7 5 Transcrição
9 3 1 7 5 Única Transcrição
3 1 9 5 7 Duplo Transcrição
3 2 9 6 7 Randômico

  - A técnica mais comum utilizada é o Módulo Aritmético. Cada posição do dígito é multiplicado através de um preso pré-definido, os produtos resultantes são adicionados e a soma é dividida pelo módulo escolhido. O dígito verificador é calculado como o módulo menos o restante do processo de divisão.

    Exemplo: Módulo 11

    - No. Conta Base

      4 5 3 6 1

    - Multiplique cada dígito escolhido pelo peso

      6 5 4 3 2

    - Produtos

      24 25 12 18 2

    - Somatório dos Produtos

      24+25+12+18+2 = 81

    - Divida a soma por 11 e retire o resto da divisão

      81/11 = 7 restou 4

    - Subtraia o resto de 11

      11-4 = 7

    - O próprio numero verificador da conta é

      45 3 61 7

    Exemplo: Módulo 10

    - No. Conta Base

      4 5 3 6 1

    - Multiplique cada dígito escolhido pelo peso

      2 0 2 0 2

    - Produtos

      8 0 6 0 2

    - Somatório dos Produtos

      8+0+6+0+2 = 16

    - Divida a soma por 10 e retire o resto da divisão

16/10 = 1 restou 6

- Subtraia o resto de 10

  10-6 = 4

- O proprio numero verificador da conta é

  4 5 3 6 1 7

- A porcentagem de erros detectados através dos módulos

  As técnicas 11 e 10 :

MÓDULOS	10	11
Transcrição	100%	94%
Única Transcrição	100%	90%
Dupla Transcrição	100%	90%
Randômico	90%	90%
Substituição de número valido porém incorreto	0%	0%

Passos no Projeto de Código

1. Descreva a informação a ser codificada.
2. Identifique o propósito do código. Determine se há um código em uso.
3. Determine o número inicial de elementos abrangidos pelo código.
4. Extrapole o número esperado de itens cobertos para os próximos 3 a 6 anos.
5. Identifique a mídia em que cada código irá aparecer como o monitor do computador, formulários e/ou relatórios.
6. Identifique o desempenhos das tarefas (identificação, classificação...) através das pessoas que estão utilizando os códigos.
7. Determine requerimentos precisos, detecção de erros e correções. Isso irá determinar se um dígito verificador é requerido.
8. Determine se há uma limitação computacional imposta no código (número de dígitos significativos suportados através da linguagem ou hardware).
9. Projete a estrutura do código.
10. Construa o código.

Figura 8 apresenta a definição de cada coluna do *The Organic Shop* e como os formatos de códigos são especificados nas tabelas.

STORE

Number	Name	Address
SMALLINT	VARCHAR(250)	VARCHAR(250)
99	X(250)	X(250)
PK,UA,ND1	**NN,ND1**	
10	GangStore in Alabama	Alabama
20	GangStore in West Virginia	West Virginia
30	GangStore in North Dakota	NorthDakota

EMPLOYEE

Number	LastName	FirstName	MiddleName	Store	Manager	Date_Hired
SMALLINT	VARCHAR(100)	VARCHAR(100)	CHAR(4)	SMALLINT	SMALLINT	DATE
9999	X(100)	X(100)	X(100)	99	9999	Mon dd, yyyy
PK,UA	**NN**	**NN**		**NN,FK**		**NN,NC**
5001	Cruz	Juan	Martinez	10	5000	25 de Dez de 2005
5002	Enriquez	Sheila	San Pedro	10	5001	4 de Jun de 2008
5003	Ferrer	Grace	Atienza	20	5000	16 de Jan de 1997
5004	Franz	Jane	Solamo	20	5003	4 de Jan de 1998

ITEM

Code	Description
SMALLINT	VARCHAR(250)
99	X(250)
PK,UA	**NN**
1001	Wheat Germs
1002	White Peppers
1003	Iodized Salts
1004	Oregano
1005	Basil Leaves

INVENTORY

Store_no	Item_code	Qty	Op_level
SMALLINT	SMALLINT	INTEGER	INTEGER
99	1XXX	99999	99999
PK1,FK	**PK1,FK**		
10	1001	50	20
20	1001	2300	500
10	1004	4500	100
20	1004	90	100

HORARIO_FUNCIONARIO

Numero	Horas_Trabalhadas
SMALLINT	DECIMAL
5XXX	999999,99
PK	
5001	250,00

SALARIO_FUNCIONARIO

Numero	Salario_Anual	Loja_opcional
SMALLINT	DECIMAL(30,2)	CHAR(1)
5XXX	999999,99	Y or N
PK		default='N'
5001	546000,00	N

CONSULTA

Numero	Num_Contrato	Comissao_Diaria
SMALLINT	VARCHAR(25)	DECIMAL(20,2)
5XXX	X(250)	999999,99
PK		
5005	AH0001-10001	750,00

Figura 8: Aparência de uma tabela de Banco de Dados de uma loja

3.2. Integridade da Entidade

Uma base de relacionamentos ou tabela base corresponde a uma entidade conceitual cujo tuplos ou filas são fisicamente armazenadas no banco de dados. A integridade da entidade significa que a base da relação da chave primária (quer sejam simples ou compostas) não pode ser nula. Por definição, uma chave primária é no mínimo um identificador utilizado para identificar linhas exclusivas. Isto significa que nenhum subconjunto da chave primária é suficiente para fornecer uma identificação única de linhas. Se *nulls* (nulo) foram autorizados, implica que nem todas as colunas são necessárias para estabelecer uma distinção entre as linhas, o que contradiz a definição da chave primária.

A integridade da entidade também é conhecido como ***primary key constraint.***

3.3. Integridade Referencial

Integridade referencial define as *constraints* que abordam a validade de referências a uma tabela para alguma outra tabela ou tabelas no banco de dados. Diz respeito à forma como são definidas as chaves estrangeiras. Se existe uma chave estrangeira em uma tabela, quer o valor chave estrangeira deve corresponder a um valor de alguns candidatos-chave na sua própria tabela ou o valor chave estrangeira deve ser nulo.

Como um exemplo, considere FUNCIONÁRIO e LOJA na tabela da Figura 1. A coluna LOJA da tabela FUNCIONÁRIO é considerado uma chave estrangeira. Os valores encontrados são os mesmos da coluna NÚMERO da tabela LOJA. Pode-se dizer que o quadro é referenciar o trabalhador, e da tabela é referenciada da loja. Cada valor na coluna na chave estrangeira referenciando a tabela deve ser o mesmo valor na chave primária correspondente coluna da tabela referenciada, ou então, deve ser nulo.

Regras referenciais de integridade

1. Regra para inserir. Esta linha afirma que não deve ser inserido na mesma tabela de referenciamento a menos que já exista uma correspondênte entrada na tabela referenciada. Considere o registro da Figura 2. Suponhamos que não haja registro em Loja Número 99 na tabela LOJA. Se essa linha é inserida, seria violar a inserção regra. A fim de permitir a inserção, quer atribuir um número que existe na tabela da loja ou atribuir um valor nulo (assumindo nulls são permitidos para esta coluna).

FUNCIONARIO

Numero	5110
Sobrenome	Salazar
Nome	Ceasar
NomeMeio	Chan
Loja	99
Gerente	5001
Data_Inicio	15/02/07

Figura 9: Funcionario 5110 gravado

2. Regra para deletar. Indica que uma linha não deve ser suprimida a partir da tabela referenciada se há alguma correspondência linhas na tabela do referenciamento. Suponhamos que uma Loja pretende apagar as informações para a Loja Número 20 (GangStore na Virgínia Ocidental). Este pedido irá violar a regra de supressão. Há três coisas que se pode fazer:

- **Restringir**. Fará com que o pedido de supressão não seja aceito ou negado. O sistema não permitirá a supressão da Loja número 20 na tabela LOJA, e apresentará um erro;
- **Anular**. Significa que os valores da chave estrangeira da tabela de referenciamento sejão fixados a um valor nulo. No exemplo, todas as linhas na tabela o empregado trabalhar na loja número 20 vai definir a sua Loja coluna nulo. Em seguida, a fila para a Loja número 20 no quadro da loja será excluído;
- **Em cascata**. Significa que as colunas na tabela de referencia serão também apagadas. No exemplo, todas as linhas na tabela FUNCIONARIO que trabalha na loja número 20 será suprimido. Em seguida, a fila para a Loja número 20 no quadro da loja será excluído.

3.4. Desencadeamento de Operações

A operação está acionando uma afirmação ou regra que rege a validade dos dados de manipulação operacionais tais como inserir, atualizar e excluir. Os componentes para desencadear uma operação são os seguintes:

- Regra de uso é uma declaração concisa do negócio regra a ser executada pelo desencadeamento operação;
- Evento que é a manipulação operação dados (inserir, atualizar ou excluir) que inicia a

operação;

- Nome da entidade que é o nome da entidade a ser acessadas ou modificadas;
- Condição que faz a operação a ser desencadeada;
- Ação que deve ser tomada quando a operação é desencadeada.

Figura 10 mostra um exemplo de como desencadear uma operação baseada na Montante em RETIRADA. Neste exemplo, se o pedido se tenta inserir uma linha na tabela a retirada no caso de o valor é mais do que o saldo da conta, o sistema irá rejeitar a operação. Não se pode retirar mais dinheiro do que aquilo que nos é encontrado em uma conta do.

```
Regra de Uso: WITHDRAWAL AMOUNT não deverá exceder ACCOUNT BALANCE
Evento: Insert
Nome da Entidade: WITHDRAWAL
Condicao: WITHDRAWAL AMOUNT > ACCOUNT BALANCE
Acao: Rejeitar a insercao de transacoes
```

Figura 10: O exemplo desencadeia a operação

4. Volume de dados e análise de uso

Uso de Dados e Análise Volume fazem parte do banco de dados físicos desenho processo. Ao realizar análises volume dados, as estimativas do tamanho do banco de dados são utilizados para selecionar física dispositivos de armazenamento e estimar o custo de armazenamento. Ao realizar análise de utilização, as estimativas de uso caminhos ou padrões são utilizados para selecionar organizações arquivo e de acesso aos métodos plano para o uso de índices e de planejar uma estratégia para a distribuição dos dados. Uma maneira fácil de mostrar as estatísticas sobre os volumes e os dados de utilização é a notação, acrescentando que a entidade-que representa a relação diagrama final conjunto de relações normalizadas a partir de dados desenho lógico.

4.1. Análise do volume de dados

Considere o diagrama entidade-relacionamento das relações da Clínica de Repouso San Nicolas Baranggay na figura abaixo.

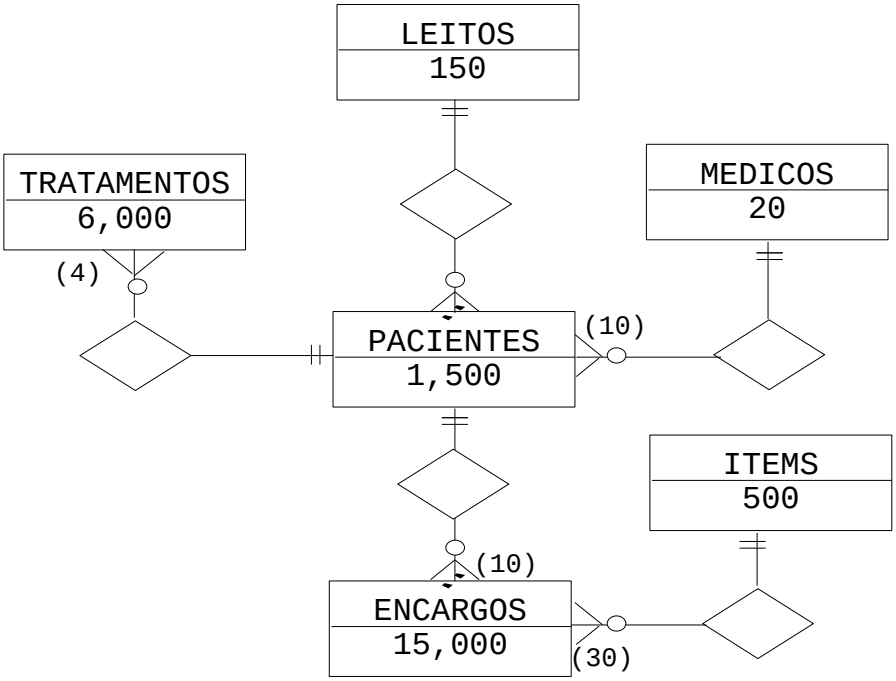


Figura 11: Clinica de Repouso San Nicolas Baranggay

Os dados relativos ao volume de registros ou filas são recolhidos, normalmente, durante a fase dos requisitos de engenharia ou na fase de análise. Números no interior das entidades serão a média do número de registros apresentados em determinado momento, relativamente. O número dentro do parênteses é o número de registro associado ao relacionamento um-para-muitos. A descrição da análise do volume de dados é a seguinte:

1. Desde que existam 150 leitos no hospital, o número máximo de pacientes admitidos em qualquer circunstância é limitado a 150
2. Em média, o registro do paciente está ativo por cerca de 30 dias
3. Em média, a duração da estadia do paciente é de 3 dias
4. O número total de pacientes ativos é de aproximadamente 1500 registros. Ele é computado como x 150 leitos (30 dias para os prontuários ativos / 3 dias de duração da estada)
5. Após o período de 30 dias, o registro do paciente é arquivado
6. A média de paciente incorre numa média de 10 cobranças durante a hospitalização. A média do número de entidades COBRANCA espera-se que seja (10 taxas x 1500 prontuários) ou 15000 registros de cobranças
7. Há 500 itens distintos que podem ser exibidos na fatura de um paciente. O número médio de cobranças pendentes para cada fatura é de(15000 cobranças/500) 30 itens
8. Cada paciente recebe uma média de 4 tratamentos. O número médio de entidades TRATAMENTOS no banco de dados é (4 tratamentos x 1500 doentes) 6000
9. Um médico examina uma média de 10 pacientes por mês

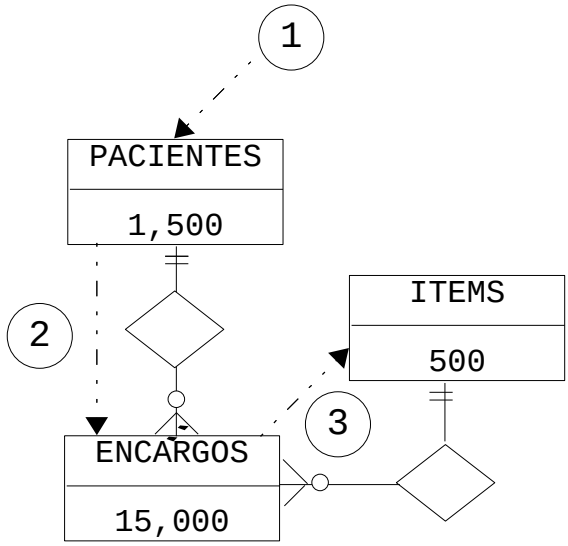
4.2. Uso da análise de dados

O analista identifica as principais transações e processamentos solicitados ao banco de dados. Cada operação de processos é então analisada para determinar os caminhos de acesso utilizados e estimar a freqüência de uso. Faça uso do transaction map (mapa de transações). Quando todas as transações foram analisadas, o composite load map é elaborado, mostrando o total de caminhos de acesso usados no modelo conceitual.

O transaction map é um diagrama que mostra a sequência lógica de acesso de dados. A Figura 5 mostra o transaction map da Clínica de Repouso San Nicolar Baranggay sempre que a fatura de um doente é criada.

TRANSACTION ANALYSIS FORM

TRANSACAO NUM: SNBLC-1 DATA: 02/20/20XX
TRANSACAO NOME: CREATE PATIENT BILL
TRANSACAO VOLUME:
MEDIA: 2 registros por hora PICO: 10 registros por hora
TRANSACTION MAP



NUM.	NOME	TIPO DE ACESSO	NUM DE REFERENCIA	
			POR TRANS.	PICO
1	ENTIDADE-PACIENTE	R	1	10
2	ENCARGOS-PACIENTE	R	10	100
3	ITEM-ENCARGOS	R	10	100

Figura 12: Criar Transaction Map do paciente Bill

1. A operação CREATE PATIENT BILL faz o registro do PACIENTE que será lido juntamente com os detalhes das despesas do paciente e também promove a impressão da fatura;
2. O volume médio de transações é de 2 por hora e 10 por hora durante o período de pico;
3. A operação exige apenas um PACIENTE como referência por transação, quando seu período de pico for 10 transações por hora;
4. Existem 10 ENCARGOS por PACIENTE. A média do número de vezes que o caminho ENCARGOS-PACIENTE é usado por transação é 10. Durante o período de pico, será 10 x 10 ou 100 por hora;
5. Uma vez que ITEM-ENCARGOS é atravessado uma vez por cada ENCARGOS, os dados de utilização também atingem um pico de utilização de 100 por hora.

O ***Composite Load Map*** é uma referência confiável para estimar o volume e o uso de dados no banco de dados.

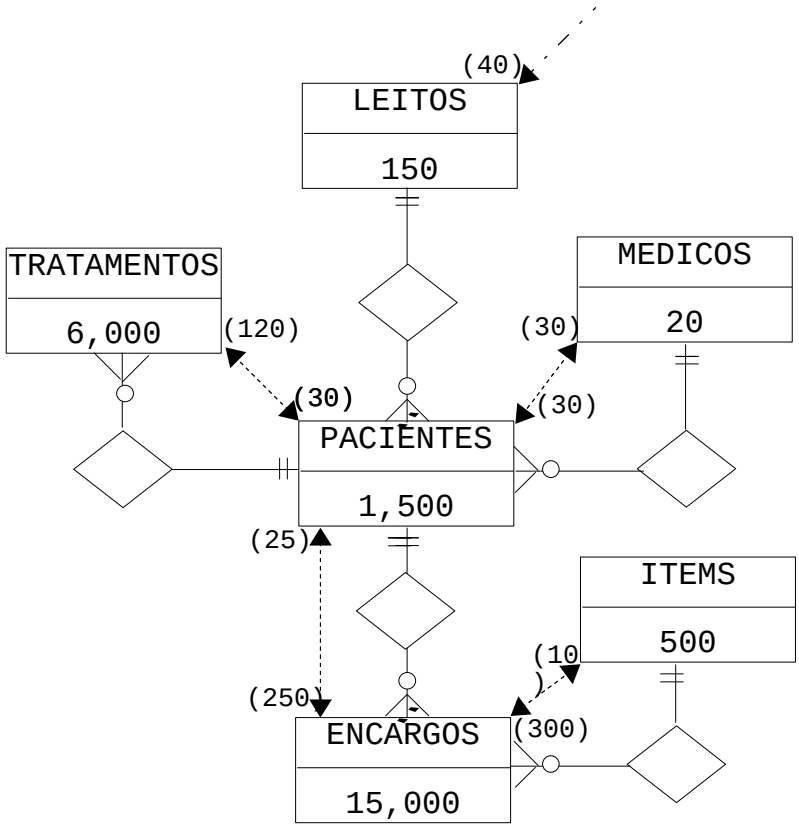


Figura 13: Composite Load Map da Clinica de Repouso San Nicolar Baranggay

1. O número à frente de cada seta tracejada é uma estimativa do número total de referências de acesso a um determinado caminho durante o pico de acessos;
2. Há registros de 40 leitos em período de pico;
3. O número de referências para a entidade a partir da entidade paciente está estimado em 120 por hora;
4. O número de referências ao paciente a partir do exterior do modelo é de 40 por hora;
5. O número de referência para os encargos da entidade o paciente entidade é estimada em 250 registros por hora;
6. O número de referências para os encargos da entidade PONTOS entidade é de 300 por hora.

Neste exemplo, nós achamos que o mais referências a entidade está ENCARGOS entidade.

5. Organização de Arquivos e Método de Acesso

Esta seção lida com os conceitos no que respeita ao armazenamento físico da base de dados secundários em dispositivos de armazenamento como discos rígidos e discos ópticos. A memória principal do computador é insuficiente para armazenar o banco de dados. Embora, por vezes o acesso primário de armazenamento são muito mais rapidamente do que armazenamento secundário, não é suficientemente grande para armazenar a quantidade de dados que pode exigir um típico banco de dados. Há alguns constrangimentos que precisa ser considerado na escolha do tipo de armazenamento secundário. São eles:

- características físicas do armazenamento secundário
- sistema operacional disponível
- software de gestão de arquivos
- necessidades do usuário para armazenamento e acesso aos dados

O banco de dados de armazenamento secundário é organizado em um ou mais arquivos, onde

cada arquivo é constituído por um ou mais registros. Cada registro terá, então, um ou mais campos. No dia anterior, um arquivo corresponde a uma tabela, um registro corresponde a uma linha e uma campo corresponde a uma coluna.

O registro físico é a unidade de transferência entre disco e memória, e vice-versa. Também é conhecida como bloco ou página, eles consistem em mais do que apenas um registro lógico. Embora, por vezes, dependendo do tamanho, um registro lógico pode corresponder a um registro físico. Hoje em dia, um banco de dados corresponde a um ou mais arquivos que podem ocupar discos inteiros; discos são divididos em páginas ou blocos.

A ordem na qual os registros são armazenados e acessos aos arquivos dependem de sua organização. ***File Organization*** (Organização de Arquivos) é uma técnica para arrumar fisicamente os registros de um arquivo em um dispositivo de armazenamento secundário. É a forma física dos dados em um arquivo de registros e páginas de armazenamento secundário. Geralmente, há duas organizações de arquivo. Sendo as seguintes:

- Sequential File Organization (organização seqüencial de arquivos)
- Hashed File Organization (organização aleatória de arquivos)

Junto com a organização de arquivos, existe um conjunto de métodos para acesso aos arquivos. O método de acesso aos arquivos define as etapas envolvidas ao armazenar e recuperar registros a partir de um arquivo. Em geral, há três métodos de acesso aos arquivos. Sendo:

- método de acesso seqüencial
- método de acesso indexado
- método de acesso aleatório ou direto

Alguns métodos de acesso podem ser aplicados apenas em determinadas organizações de arquivos. Por exemplo, o método de acesso aleatório não pode ser aplicado na organização seqüencial de arquivos.

Veremos a seguir os critérios para selecionar uma organização de arquivos:

1. rapidez na recuperação
2. agilidade no processamento de transações
3. utilização eficiente de espaço de armazenamento
4. proteção para falhas ou perda de dados
5. diminuir a necessidade de reorganização
6. possibilitar o crescimento
7. segurança, para evitar uso não autorizado

5.1. Organização Seqüencial de Arquivos

A organização de arquivos seqüencial é o tipo mais comum de organização de arquivos. Os registros são armazenados de forma a registar o campo-chave. Outro nome de arquivo seqüencial é pilha. Como exemplo, considere que a Figura 7 mostra a forma como os registros da tabela LOJA são armazenados em disco. Elas são organizadas de acordo com seu número, onde o menor número da loja é o primeiro da lista. Os métodos de acesso aos arquivos, que podem ser usados para recuperar registros em um arquivo seqüencial, são os seguintes:

Método de Acesso Seqüencial

Para localizar um determinado registro, o usuário normalmente procura o arquivo a partir do início até onde o registro está localizado. Isto, torna a recuperações de registros relativamente lenta. Para inserir ou atualizar um registro, é necessário ir até o arquivo. Porque os registros não podem ser inseridos no meio do arquivo, um arquivo seqüencial é normalmente copiado durante a atualização em processo.

Para eliminar um registro, sua primeira página deve ser recuperada, o registro é marcado como apagado. Quando a página é escrita de volta no disco, seu espaço é automaticamente excluído dos registros e reutilizado. Os arquivos são periodicamente reorganizados pelo administrador do banco de dados, para recuperar o espaço disponibilizado por registros suprimidos.

Método Indexado de Acesso aos Arquivos

Discutido na seção de índices desta lição.

LOJA

Numero	Nome	Endereço
10	GangStore in Alabama	Alabama
20	GangStore in West Virginia	West Virginia
30	GangStore in North Dakota	NorthDakota
40	GangStore in Michigan	Detroit, Michigan
50	GangStore in South Carolina	South Carolina

Figura 14: Exemplo de arquivo para armazenar a tabela seqüencial

5.2. Hashed File Organization

Outra organização de arquivos é a organização de arquivo *hash,* onde os registros são armazenados no disco aleatoriamente. O endereço para cada registro é determinada usando um algoritmo *hash*. Algoritmo *hash* ou função *hash* é uma rotina que calcula o endereço do registro. O campo *hash* é o domínio no qual a base calcula o endereço. Normalmente, esta é a principal chave do registro. Neste caso, a área *hash* é conhecida como a chave *hash*. Registros dentro de um arquivo *hash* aparecem aleatoriamente distribuídos em todo o espaço disponível em disco. Por esta razão, os arquivos *hash* são muitas vezes chamado aleatórios ou arquivos diretos.

Um método comum utilizado para o algoritmo ou função *hash* é o restante da divisão. Dividir o número original por um número primo que se aproxima lda quantidade a ser armazenada. Em seguida, o resto da divisão é usado como o endereço. Por exemplo, suponha que uma empresa tem 1000 funcionários registrados. Uma boa aproximação de valor seria 997, porque é perto de 1000. Pense em um registro tendo a chave primária 50542. 50542 dividido por 997. O restante é 629. Esta é a localização (endereço da página ou bloco) do registro no disco.

Algoritmos ou funções *hash* não garantem um endereço exclusivo, porque o número de possíveis valores *hash* que um campo pode ter, geralmente, é muito maior do que o número de endereços disponíveis para os registros.

Cada endereço gerado a partir de uma função *hash* corresponde a uma página com saídas para vários registros. Dentro da página, os registros são inseridos em sua ordem de chegada. Quando o mesmo endereço é gerado por dois ou mais campos *hashdiferentes*, dizemos que ocorreu um conflito. Quando isso acontece, o novo registro deve ser inserido em outro local. Gerenciar conflitos complica a gestão de arquivos *hash* e degrada o desempenho global. Existem várias técnicas que podem ser usadas para gerenciar conflitos:

1. Gerando endereços. Quando há um conflito, o registro é inserido no primeiro endereço disponível. Quando a última página foi pesquisada e não existem faixas disponíveis, o sistema inicia uma nova página. Consideremos o exemplo na figura 8, onde uma página pode conter um máximo dois registos (duas faixas disponíveis). Suponhamos que temos de inserir o registro para a Loja 50 na tabela LOJA. Suponhamos também, que quando o algoritmo *hash* foi utilizado, ele enviou para o endereço da página 1. Note que não há mais faixas disponível. O sistema irá pesquisar linearmente a primeira faixa que estiver disponível na página 2 faixa 1;

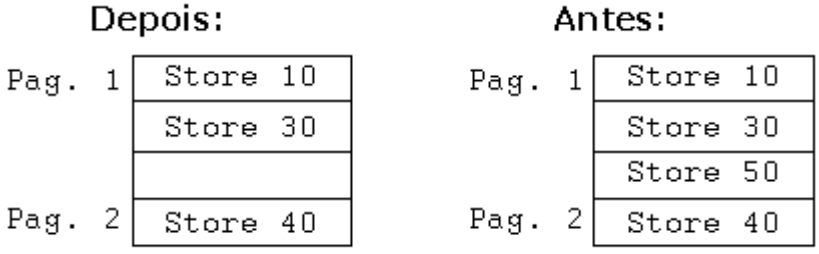


Figura 15: Gerando Endereços

2. Estouro de memória desencadeado. Quando há um colisão, um transbordamento é mantida área que não pode ser colocada no endereço *hash.* O registro é inserido na área

de overflow. Considere o exemplo mostrado na Figura 9. Suponhamos que temos de inserir o registro para a Loja 50 na tabela LOJA. Suponhamos também, que quando o algoritmo *hash* é usado, ele levou para o endereço da página 1. Note que não há mais slot disponível. O sistema irá colocar o registro no overflow. Neste exemplo, é Pag. 2 faixa 1.

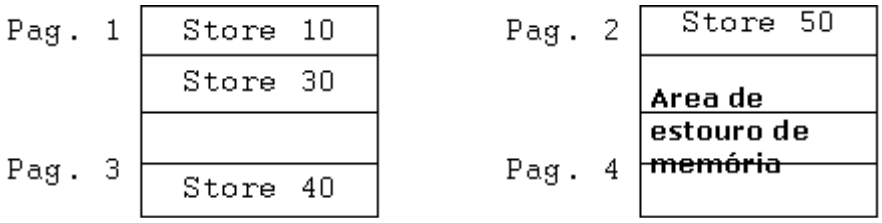


Figura 16: Estouro de memória

3. Estouro de memória encadeado. Esta é uma variação do estouro de memória desencadeado. Uma área de estouro de memória é mantida para serem colocados os endereço *hash*. Uma outra área é usada pela página. O campo, também conhecido como sinônimo de ponteiro e indica se um conflito ocorreu. Pag. 1 aponta para a área que está sobrando Pag. 2. Pag. 3 não tem nenhum espaço sobrando.

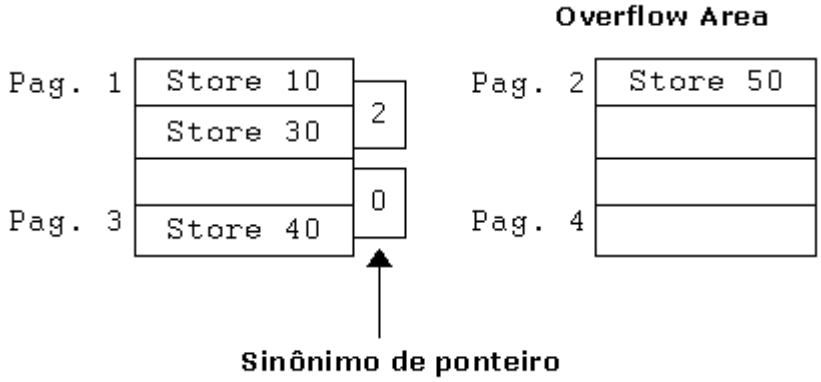


Figura 17: Estouro de memória

4. Múltiplos hashes. Quando um conflito ocorre, uma segunda função hash é utilizada. O objetivo é produzir um novo endereço hash que permitirá evitar o conflito. A segunda função hash é usada para fazer registros em uma área de estouro de memória.

6. Índices

Nesta seção, serão discutidas técnicas para fazer o retorno de dados mais eficiente usando índices. Um índice é uma tabela ou outra estrutura de dado que é usada para determinar a localização de linhas em uma tabela (ou tabelas) que satisfaçam alguma condição. Pode ser definido como chave primária e valores de atributo sem chave. São mais compactos que os registros de dados (ou linhas) a que fazem referência. Podem ser alocados em memória durante períodos longos a fim de reduzir o custo de acesso a memória secundária para retorná-los.

Uma estrutura de índice é associada com uma chave especial de procura, e contém registros que consistem do valor da chave e o endereço do registro lógico no arquivo que contém o valor da chave. O arquivo de dado é um arquivo contendo os registros lógicos enquanto que o arquivo de índice contém os registros de índice. Os registros no arquivo de índice são ordenados de acordo com o campo de indexação, que é geralmente baseado em uma coluna.

Quando usamos índices?

- Devem ser generosamente para aplicações que são usadas primeiramente para suportar retorno de dados como as aplicações de suporte a decisão.
- A contra-partida entre a performance melhorada através do uso de índices e a queda de performance pelos métodos de inserção, deleção, e atualização de registros deve ser compreendida. A performance cai quando se modifica o conteúdo da tabela porque também se precisa atualizar os índices das tabelas.

6.1. Arquivos Seqüencialmente Indexados

Arquivos de dados armazenam registros seqüencialmente ou de forma não-seqüencial, e um índice é criado para permitir que as aplicações localizem registros individualmente. Existem dois tipos básicos de organização de arquivos por índice. São eles:

Organização de arquivos com índice em seqüência: Os registros são armazenados seqüencialmente pelo valor da chave primária. Um índice simples, chamado índice de bloco ou índice primário, pode ser usado. Esta estrutura é um compromisso entre um arquivo puramente seqüencial e um arquivo puramente aleatório, podendo ser processada seqüencialmente ou individualmente acessada usando uma chave de procura que acessa o registro pelo índice. Uma organização de arquivo com índice em seqüência, normalmente, tem:

- uma chave primária
- um índice ou índices separados
- uma área de overflow (estouro)

Um método largamente usado é o Método de Acesso Seqüencial-Indexado (ISAM) definido pela IBM. Um índice é dito esparso se somente alguns valores da chave de busca tem índices. Um índice é dito denso se cada um dos valores da chave busca tem índices. A seguinte figura ilustra isso:

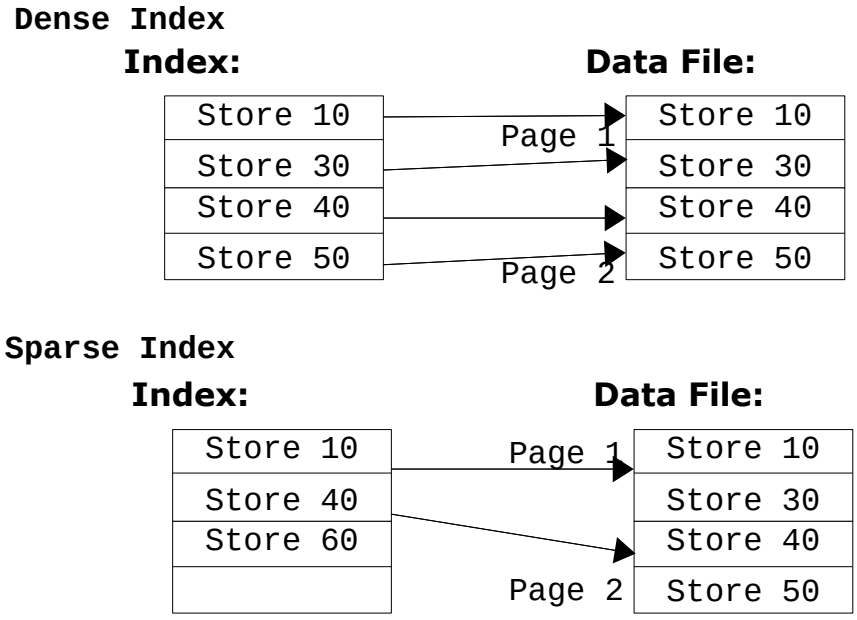


Figure 18: Dense vs. Sparse Index

Um exemplo de organização de arquivos com índice em seqüência, é uma lista de telefones. No canto esquerdo da página, pode-se achar o último nome da primeira pessoa daquela página. No canto direito, pode-se achar o último nome da última pessoa daquela página. Outro exemplo é ilustrado na figura 19. Aqui, os registros da tabela **STORE** (arquivo de dados) são guardados em forma seqüencial de acordo com a chave primária NUMBER. O STORE-ADDRESS-INDEX **(arquivo índice)** contém os endereços organizados em ordem alfabética, e cada endereço aponta para o registro na tabela STORE.

STORE-ADDRESS-INDEX

Alabama	1
Detroit, Michigan	4
North Dakota	3
South Carolina	5
West Virgina	4

STORE

	Number	Name	Address
1	10	GangStore in Alabama	Alabama
2	20	GangStore in West Virginia	West Virginia
3	30	GangStore in North Dakota	NorthDakota
4	40	GangStore in Michigan	Detroit, Michigan
5	50	GangStore in South Carolina	South Carolina

Figure 19: Exemplo Indexed File Organization

Organização de Arquivos com Índices Não-Seqüenciais: Os registros são guardados não-seqüencialmente, um índice completo, chamado índice inverso, é necessário. Um exemplo são os livros em uma livraria, onde são guardados de acordo com o número do catálogo, e não de acordo com o nome do autor e título. Os catálogos de título e autor são índices completos que permitem uma pessoa encontrar rapidamente a posição correta de um certo livro. Outro exemplo é mostrado na figura a seguir, os dados na tabela STORE não são armazenados de acordo com a chave primária NUMBER. O STORE-ADDRESS-INDEX (índice do arquivo) contém o endereço dos registros organizados em ordem alfabética, e cada endereço aponta para o registro na tabela STORE.

STORE-ADDRESS-INDEX

Alabama	4
Detroit, Michigan	3
North Dakota	1
South Carolina	5
West Virgina	2

STORE

	Number	Name	Address
1	30	GangStore in North Dakota	North Dakota
2	20	GangStore in West Virginia	West Virginia
3	40	GangStore in Michigan	Detroit, Michigan
4	10	GangStore in Alabama	Alabama
5	50	GangStore in South Carolina	South Carolina

Figure 20: Indexed Non-sequential File Organization

6.2. Índices em Cluster

Se o campo indexado não é um campo chave da tabela, então pode ser mais que um registro correspondendo ao mesmo valor no campo indexado, o índice é chamado de índice em cluster ou índice secundário. São índices baseados em atributos não-chave. Normalmente, são usados quando registros em um arquivo são retornados frequentemente baseados nestes valores. O propósito é aumentar a velocidade das respostas retornadas ordenando fisicamente os registros do índice do arquivo naquele atributo não-chave, chamado atributo em cluster, que é usado para juntar ou agrupar linhas que tem um valor em comum para esse atributo. Há várias técnicas para lidar com índices secundários não-únicos.

1. Criar um índice secundário grande que mapeia para todos os registros no arquivo de dados permitindo que valores duplicados de chaves apareceram no índice
2. Ter um índice secundário para ter uma entrada no índice para cada valor de chave distinto e ter os apontadores de bloco sendo multi-valorados com uma entrada correspondente a cada valor duplicado de chave no arquivo de dados
3. Ter um índice secundário para ter uma entrada no índice para cada valor de chave distinto mas o ponteiro de bloco não apontaria para o arquivo de dados e sim para uma página que contém ponteiros para os registros correspondentes no arquivo de dados

A próxima figura mostra um exemplo de um arquivo de índice baseado em atributos não-chave. O DESCRIPTION-INDEX é um arquivo de índice para a coluna DESCRIPTION da tabela PRODUCT. Quando há o retorno de registros da tabela PRODUCT baseado na descrição do produto tal que a busca por todos os produtos com a descrição de DESK, o arquivo de índice DESCRIPTION-INDEX é primeiro procurado por todos os valores que são iguais aos da busca. Veja, que o arquivo de índice mostra que os registros 5 e 9 são produtos tendo DESK como descrição.

DESCRIPTION-INDEX

DESCRIPTION	RECORD NO.
Chair	4,6
Book Shelf	8
Desk	5,9
Stand	7

PRODUCT

	Number	Description	...
4	1000	Chair	
5	3500	Desk	
6	6250	Chair	...
7	9750	Stand	
8	1250	Book Shelf	...
9	1425	Desk	

Figura 21: Índices para os atributos não-chaves

6.3. Índices de Multi-Nível

Quando um arquivo de índice torna-se grande e se estende para várias páginas, o tempo necessário para buscar os registros aumenta. Um índice multi-nível tenta reduzir o alcance das buscas. Ele faz isso tratando o índice como um arquivo qualquer, divide o índice em outros índices menores e mentém um índice para esses outros índices. A figura 21 mostra um exemplo. Cada página no arquivo de dados armazena dois registros; na verdade, uma página compõe mais registros. Cada registro de índice armazena um valor de chave de acesso e um endereço de página. O valor chave armazenado é o maior na página endereçada.

Para localizar um registro, se inicia pelo índice de primeiro nível e procura-se na página pelo último acesso ao valor da chave que é menor ou igual à chave do registro. Por exemplo, procuramos a STORE 39, pelo índice de primeiro nível, nós achamos uma chave que é menor ou igual a este valor, nesse caso, STORE 37. Esse registro contém o endereço para o índice de segundo nível da página. Para continuar a busca, nós achamos uma chave que é menor ou igual a chave dos registros. Nesse caso, é a STORE 37 que contém o endereço da página onde a STORE 39 está localizada. Repetindo o processo chegamos a terceira página (terceiro nível).

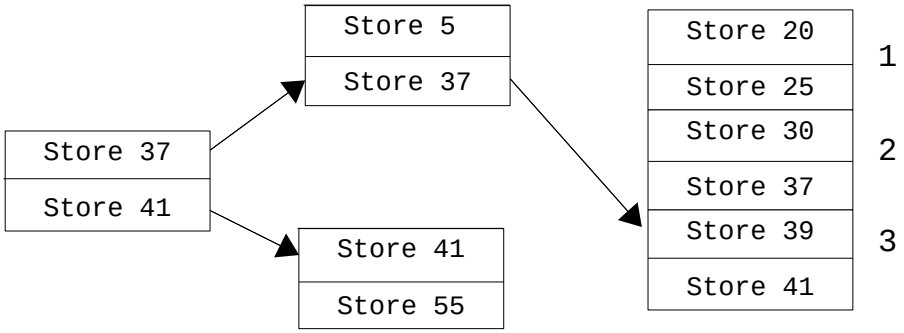


Figura 22: Multi-level Indexes

6.4. Índices Árvores

Uma árvore é uma estrutura de dados que consiste de um grupo de nós que saem de um outro nó. Isso forma uma árvore de cabeça para baixo. Consiste do seguinte:

1. **Nó raiz**: é o nó no topo da árvore
2. **Nó folha**: é um nó em uma árvore que contém um nó filho
3. **Subárvore**: é um nó e todos os descendentes daquele nó
4. **Ponteiro**: é um campo contendo dados que podem ser usados para achar um registro relacionado

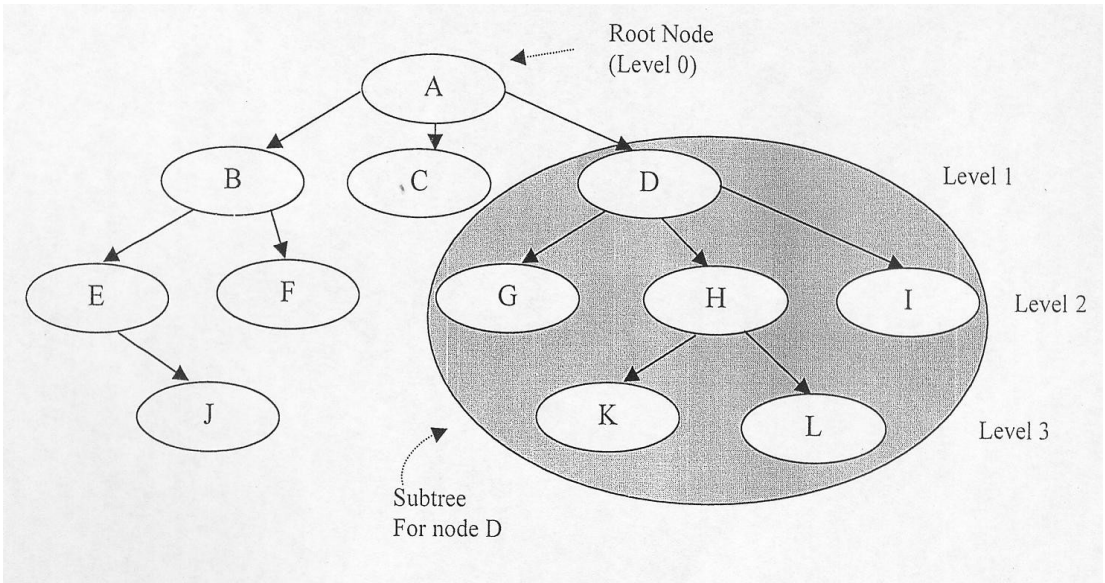


Figura 23: Estrutura de Árvore

Uma árvore tem três propriedades a seguir:

1. **Acessibilidade Uniforme**: é alcançada quando todos os nós folha têm a mesma distância da raiz. Os registros acessados frequentemente são colocados perto do nó raiz para se ter menos acesso ao disco. O acesso ao disco ocorre quando se passa entre níveis.
2. **Fator de bifurcação**: é também conhecido como o grau da árvore. É o número máximo de nós filhos permitido por um nó pai. Uma grande bifurcação geral, em geral, mais árvores superficiais.
3. **Profundidade**: é o número de níveis entre o nó raiz e um nó folha na árvore. Quando a profundidade é a mesma do nó raiz para cada nó folha, a árvore é dita balanceada.

Árvores B+ são árvores onde todas as folhas estão a uma mesma distância da raiz. É uma árvore binária de ordem 2 onde cada nó não tem mais do que dois filhos. As regras para uma árvore B+ são as seguintes:

1. Se a raiz não é um nó folha, deve ter ao menos dois filhos.
2. Para uma árvore de ordem n, cada nó, exceto a raiz e os nós folha, deve ter entre n/2 e n ponteiros e filhos. Se n/2 não for inteiro, é arredondado.
3. Para uma árvore de ordem n, o número de valores chave em um nó folha deve ser entre (n-1)/2 e (n-1) ponteiros e filhos. Se (n-1)/2 não for inteiro, o resultado é arredondado.
4. O número de valores chave contidos em um nó não-folha é 1 número menor do que o número de ponteiros.
5. A árvore deve sempre estar balanceada, ou seja, cada caminho do nó raiz para um nó folha deve ter o mesmo tamanho.
6. Nós folha são ligados em ordem de valores de chave.

A figura 23 mostra um exemplo de uma árvore B+.

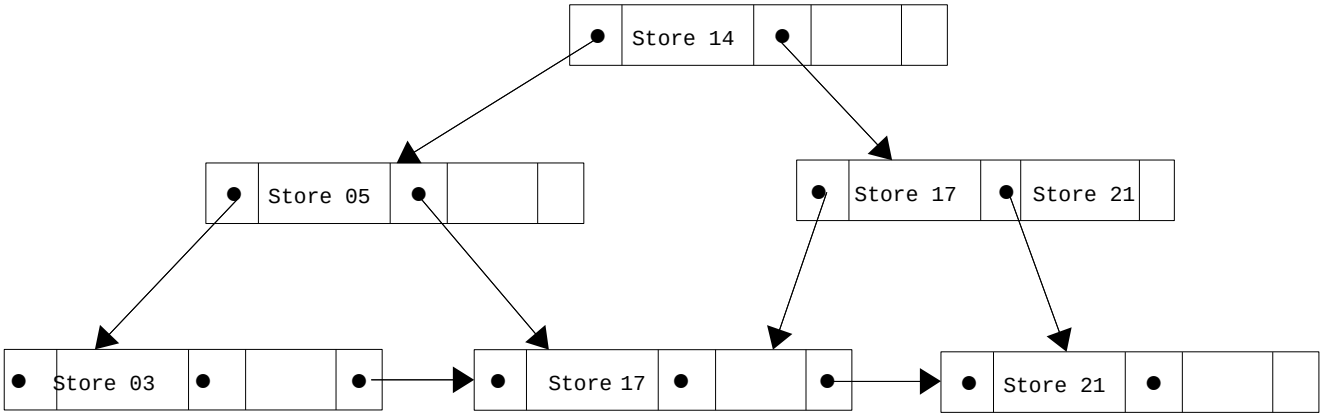
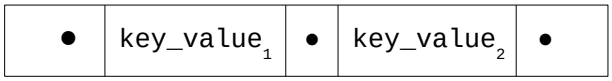


Figura 24: B+Tree Structure

Cada nó é da forma abaixo:



Onde • representa um ponteiro para outro registro ou vazio (ponteiro para lugar nenhum). Se o valor da chave de busca é menor ou igual a key_value_i, o ponteiro a esquerda de [] é usado para achar o próximo nó a ser procurado. Senão, o ponteiro para o fim do nó é usado. Por exemplo, queremos achar o registro chamado STORE 21. A busca começa do nó raiz, STORE 21 é maior que STORE 14. Seguimos o ponteiro para a direita, o que nos leva ao nó de segundo nível contendo os valores STORE 17 e STORE 21. Continuamos seguindo o ponteiro pela esquerda o que nos leva ao registro de STORE 21.

6.5. Desnormalização

Desnormalização é o processo de transformar tabelas normalizadas em especificações de registros

físicos anormalizados. Em geral, a desnormalização deve:

- Combinar colunas de várias tabelas para formar um registro físico.
- Particionar uma tabela em vários registros físicos.
- Fazer uma combinação das duas opções acima.

Um cuidado deve ser tomado quando for desnormalizar tabelas porque aumenta a chance de erros ou inconsistências. Às vezes, pode até forçar a reprogramação de aplicações devido a mudança das regras de negócio. Pode otimizar certos processamentos de dados ao custo de outras operações.

Abaixo uma lista de situações em que comumente a desnormalização é considerada:

1. **Duas entidades com um relacionamento um-para-um**. Mesmo se uma das entidades for um participante opcional, se a entidade que é procurada existe a maior parte do tempo, então talvez seja inteligente combinar estas duas entidades em uma tabela.
2. **Um relacionamento muitos-para-muitos com atributos não-chave**. Ao invés de causar a junção de tabelas de árvore para extrair dados das duas entidades no relacionamento, avisará para combinar colunas de uma das entidades dentro da tabela representando o relacionamento muitos-para-muitos e evitando uma junção em muitas pesquisas.
3. **Referência a dados**. Dados referenciados existem em uma entidade do lado-um num relacionamento um-para-muitos, e essa entidade não participa em nenhum outro relacionamento de base de dados.

A maioria das oportunidades mostradas acima concordam com a combinação de tabelas para minimizar a operação de junção. A desnormalização pode ser usada para criar mais divisão ou particionamento de tabelas em muitas outras tabelas. O particionamento pode ser:

- **Particionamento Horizontal**: quebra uma tabela em múltiplas especificações de registros colocando linhas diferentes em diferentes tabelas baseado em valores comuns. Considere o inventário para cada loja em THE ORGANIC SHOP. A tabela INVENTORY pode ser dividida baseando-se no valor da coluna STORE_NO desde que o processamento do inventário de uma loja é feito separadamente de outra loja. Veja a figura 24 para a ilustração de um particionamento horizontal. Observamos que cada tabela teria a mesma estrutura. Cada linha dentro de cada tabela são registros para uma loja específica, ou seja, INVENTORY_10 conterá apenas registros para STORE No. 10, INVENTORY_20 só conterá registros para STORE No 20, etc, INVENTORY_10 e INVENTORY_20 são chamados de uma partição da tabela INVENTORY.

INVENTORY

Store_No	Item_Code	Count	Operational Level
10	1001	50	20
20	1001	2300	500
10	1004	4500	100
20	1004	90	100

AFTER HORIZONTAL PARTITIONING

INVENTORY_10

Store_No	Item_Code	Count	Operational Level
10	1001	50	20
10	1004	4500	100

INVENTORY_20

Store_No	Item_Code	Count	Operational Level
20	1001	2300	500
20	1004	90	100

Figura 25: Particionamento Horizontal

Alguns detalhes importantes sobre o Particionamento Horizontal:

1. Faz sentido quando categorias diferentes de linhas de uma tabela são processadas separadamente. No exemplo, há o processamento do inventario da loja numero 10 separadamente do inventario da loja 20.
2. Pode ser seguro porque podemos limitar o acesso de usuários a certas linhas de dados. Empregados da loja numero 10 so podem acessar INVENTORY_10; não podendo acessar INVENTORY_20.
3. Pode permitir deixar uma partição 'sem serviço' porque ela foi danificada ou então ela pode ser recuperada permitindo ainda o processamento de outras partições enquanto o problema persistir.
4. Pode permitir partições diferentes residirem em discos físicos diferentes.

- **Particionamento Vertical**: é a distribuição das colunas de uma tabela em muitas tabelas separadas. Como um exemplo, supondo queremos que apenas certas aplicações possam acessar e modificar a coluna "OPERATIONAL LEVEL"; outras aplicações acessam e modificam a coluna "COUNT". Isto é por propósitos de segurança. Podemos dividir a tabela INVENTORY como mostrado na Figura 25. A coluna INVENTORY_COUNT pode ser acessada por aplicações que gerenciam reservas e vendas de estoques enquanto a INVENTORY_LEVEL pode ser acessada por aplicações que gerenciam compras e manutenções. INVENTORY_COUNT e INVENTORY_LEVEL são as partições.

INVENTORY

Store_No	Item_Code	Count	Operational Level
10	1001	50	20
20	1001	2300	500
10	1004	4500	100
20	1004	90	100

AFTER VERTICAL PARTITIONING

INVENTORY_COUNT

Store_No	Item_Code	Count
10	1001	50
10	1004	4500

INVENTORY_LEVEL

Store_No	Item_Code	Level
20	1001	500
20	1004	100

Figura 26: Particionamento Vertical

Combinações de particionamento vertical e horizontal são possíveis. Essa forma de desnormalização é comum para uma base de dados que é distribuída entre muitos computadores. A discussão de base de dados distribuída está além do escopo desse curso.

Uma outra forma de desnormalização é a replicação de dados onde os mesmos dados estão propositalmente armazenados em lugares múltiplos. A discussão a respeito de replicação de dados está fora do escopo desse curso.

7. Exercícios

Desenvolver o modelo físico da base de dados a partir do modelo lógico desenvolvido do problema 3 na lição 3 (o gerenciador de apartamento). Defina o seguinte:

- Restrições de domínio
- Integridade da entidade
- Integridade referencial
- Operações com gatilho

Desenvolver o modelo físico da base de dados gerada do problema 4 na lição 3 (o gerenciador de loja de eletrônico).

- Restrições de domínio
- Integridade da entidade
- Integridade referencial
- Operações com gatilho

Desenvolver o modelo físico da base de dados desenvolvida para o sistema de distribuição e-Lagyan.

- Restrições de domínio
- Integridade da entidade
- Integridade referencial
- Operações com gatilho

Parceiros que tornaram JEDI™ possível

Instituto CTS
Patrocinador do DFJUG.

Sun Microsystems
Fornecimento de servidor de dados para o armazenamento dos vídeo-aulas.

Java Research and Development Center da Universidade das Filipinas
Criador da Iniciativa JEDI™.

DFJUG
Detentor dos direitos do JEDI™ nos países de língua portuguesa.

Banco do Brasil
Disponibilização de seus *telecentros* para abrigar e difundir a Iniciativa JEDI™.

Politec
Suporte e apoio financeiro e logístico a todo o processo.

Borland
Apoio internacional para que possamos alcançar os outros países de língua portuguesa.

Instituto Gaudium/CNBB
Fornecimento da sua infra-estrutura de hardware de seus servidores para que os milhares de alunos possam acessar o material do curso simultaneamente.