# Parte 3 – Práticas de Contagem

Página intencionalmente deixada em branco.

# Parte 3 Práticas de Contagem

**Introdução**

A parte 3 fornece práticas de contagem detalhadas e exemplos aperfeiçoados para apoiar os praticantes no entendimento de:

- Dados de código, incluindo uma descrição dos vários tipos de dados, como distinguir dados de negócio e de referência de dados de código e como considerar os dados de código
- Arquivos lógicos, incluindo uma descrição de arquivos lógicos e o processo para agrupar corretamente dados em arquivos lógicos; para contar corretamente tipos de registro elementares; e para contar corretamente tipos de dados elementares
- Dados compartilhados, incluindo uma descrição de dados compartilhados e o processo para identificar quando dados compartilhados são funções to tipo dado (arquivo lógico interno, arquivo de interface externa) ou funções do tipo transação (entrada externa, saída externa, consulta externa)
- Projetos de melhoria e atividades de manutenção, incluindo descrições de melhoria e manutenção e aquelas atividades reconhecidas pelo processo de contagem de pontos de função

**Conteúdo**

A parte 3 inclui os seguintes capítulos:

Esta página foi intencionalmente deixada em branco.

# Parte 3 Capítulo 1

# Dados de Código

**Introdução**

Este capítulo usa o conceito de requisitos funcionais e não-funcionais de usuário (descritos na Parte 1 e em "A Framework for Functional Sizing" [IFPUG, 2003]) para identificar dados de código e determinar como devem ser considerados.

Este capítulo aborda especificamente um número de exemplos relevantes para dados de código. É reconhecido que pode haver outros exemplos de dados de código e esses podem ser abordados em futuras versões do CPM.

**Conteúdo**

Este capítulo inclui as seguintes seções:

# Tipos de Entidades de Dado

Uma revisão dos dados da aplicação e de seu propósito fornece um entendimento das várias categorias de entidade de dado. Em geral, os analistas devem distinguir entre três categorias de entidades de dado:

- Dados de Negócio
- Dados de Referência
- Dados de Código

As primeiras duas categorias de entidades usualmente são identificadas para satisfazer os Requisitos Funcionais do Usuário e dessa forma serão investigadas para contagem como arquivos lógicos (veja Parte 3, Capítulo 2).

A terceira categoria de dados, referenciada a seguir neste capítulo como "Dados de Códigos", contudo, geralmente existe para satisfazer requisitos não-funcionais do usuário (para requisitos de qualidade, implementação física e/ou uma razão técnica) ao invés de um requisito funcional do usuário. As diferentes categorias de dados são delineadas abaixo para apoiar na identificação.

## Dados de Negócio

Dados de Negócio também podem ser chamados Dados Essenciais do Usuário ("Core User Data") ou Objetos de Negócio. Este tipo de dado reflete a informação necessária a ser armazenada e recuperada pela área funcional abordada pela aplicação. Dados de Negócio geralmente representam um percentual significativo das entidades identificadas. Possuem a maioria das seguintes características:

### Lógicas

Características lógicas incluem:

- Obrigatória para a operação da área funcional do usuário
- Identificável pelo usuário (geralmente por um usuário do negócio)
- Capaz de ser mantida pelo usuário (geralmente um usuário do negócio)
- Armazena os Dados Principais do Usuário para apoiar transações de negócio
- Muito dinâmica – as operações normais de negócio fazem com que sejam regularmente referenciados e rotineiramente incluídos, alterados ou excluídos
- Capaz de ser reportada

**Físicas** Características físicas incluem:

- Possui campos-chave e geralmente vários atributos
- Pode ter de zero a uma infinidade de registros

**Exemplos** Exemplos de Dados de Negócio incluem:

- Arquivo de Cliente, Arquivo de Fatura, Arquivo de Empregado, Arquivo de Função
- O Arquivo de Função, no Sistema de Gerência de Funções, incluiria itens como:
  - Número da Função,
  - Nome da Função,
  - Nome da Divisão,
  - Data de Ativação da Função, etc.

## Dados de Referência

Este tipo de dado é armazenado para apoiar as regras de negócio na manutenção de dados de negócio; por exemplo, em uma aplicação de folha de pagamento ele seria o dado armazenado sobre as alíquotas de impostos do governo para cada faixa salarial e a data em que a mesma iniciou a sua vigência. Os Dados de Referência geralmente representam um pequeno percentual das entidades identificadas. Possuem a maioria das seguintes características:

**Lógicas** Características lógicas incluem:

- Obrigatório para a operação da área funcional do usuário
- Identificável pelo usuário (geralmente um usuário de negócio)
- Geralmente capaz de ser mantido pelo usuário (Geralmente por um usuário administrativo)
- Geralmente estabelecido quando a aplicação é instalada pela primeira vez e mantido intermitentemente
- Armazena os dados para apoiar as principais atividades do usuário
- Menos dinâmico – ocasionalmente muda em resposta às mudanças no ambiente da área funcional, processos funcionais externos e/ou regras de negócio
- Transações processando Dados de Negócio costumam precisar de acesso a Dados de Referência

**Fisicas** Características físicas incluem:

- Possui campos-chave e poucos atributos
- Geralmente pelo menos um registro ou um número limitado de registros

**Exemplos** Exemplos de Dados de Referência incluem:

- Faixas Salariais, Taxas de Desconto, Alíquotas de Impostos, Configuração de Limites
- Arquivo de Faixas Salariais – armazena informação sobre os valores pagos para cada tipo de função e a habilidade exigida para executar aquele tipo de função
  - Tipo de Função
  - Situação, Valor Cobrado, Inicio de Vigência (1:n)
  - Descrição das Habilidades da Função (1:n)

## Dados de Código

O usuário nem sempre especifica diretamente os Dados de Código, às vezes chamados Dados de Lista ou Dados de Tradução. Em outros casos são identificados pelo desenvolvedor em resposta a um ou mais requisitos não-funcionais do usuário. Os Dados de Código fornecem uma lista de valores válidos que um atributo descritivo pode ter. Normalmente os atributos de Dados de Código são Código, Descrição e/ou outros atributos 'padrão' descrevendo o código; por exemplo, abreviação padrão, data de início de vigência, data de expiração, dados de trilha de auditoria, etc.

Ao utilizar códigos em Dados de Negócio, é necessário ter meios de tradução para converter de código para algo mais reconhecível pelo usuário. De modo a satisfazer os requisitos não-funcionais do usuário, os desenvolvedores quase sempre criam uma ou mais tabelas contendo Dados de Código. Logicamente, o código e a sua descrição correspondente têm o mesmo significado. Sem uma descrição, o código nem sempre pode ser claramente entendido.

A diferença chave entre Dados de Código e Dados de Referência é:

- Com os Dados de Código, você pode substituir um pelo outro sem mudar o significado dos Dados de Negócio; por exemplo, Código do Aeroporto versus Nome do Aeroporto, Id da Cor versus Descrição da Cor.
- Com Dados de Referência, você não pode substituir (por exemplo, Código do Imposto pela Alíquota do Imposto).

Os Dados de Código possuem a maioria das seguintes características:

**Lógicas**

Características lógicas incluem:

- O dado é obrigatório para a área funcional, mas é opcionalmente armazenado como um arquivo de dados
- Não é geralmente identificado como parte dos requisites funcionais do usuário; é geralmente identificado como parte da solução para atender requisitos não-funcionais do usuário
- É algumas vezes mantido pelo usuário (geralmente por uma pessoa de suporte ao usuário)
- Armazena dados para padronizar e facilitar atividades e transações de negócio
- É essencialmente estático – apenas muda em resposta a mudanças na forma como o negócio funciona
- As transações de negócio referenciam os Dados de Código para melhorar a facilidade da entrada de dados, melhorar a consistência dos dados, garantir a integridade dos dados, etc.
- Se reconhecido pelo usuário:
  - Algumas vezes é considerado como um grupo do mesmo tipo de dados
  - Pode ser mantido usando a mesma lógica de processamento

**Físicas**

Características físicas incluem:

- Consiste de campo-chave e geralmente apenas um ou dois atributos
- Normalmente possui um número estável de registros
- Pode representar 50% de todas as entidades em 3ª Forma Normal
- É às vezes desnormalizado e colocado em uma tabela física com outros Dados de Código
- Pode ser implementado sob diferentes formas (por exemplo, via aplicação distinta, dicionário de dados ou hard-coded dentro do software)

**Exemplos**

Exemplos de Dados de Código incluem:

- Estado
  - Código do Estado
  - Nome do Estado
- Tipo de Pagamento
  - Código do Tipo de Pagamento
  - Descrição do Pagamento

## Origem dos Dados de Código

Historicamente, a motivação para os dados de código foi economizar espaço por meio do armazenamento de um código ao invés de uma longa descrição textual. Para facilidade de manutenção, esses códigos e descrições eram colocados em arquivos ou tabelas a fim de eliminar mudanças no software quando atualizações fossem necessárias.

Os Dados de Código são uma propriedade de um atributo descritivo chamado "Meta Dado". Exemplos são valores válidos, descrições de códigos ou tabelas de tradução. Alguns Dados de Código são desenvolvidos para atender requisitos específicos do usuário e contém dados que estão dentro do domínio do usuário. Outros Dados de Código podem ser derivados a partir dos requisitos do usuário para restringir os valores permitidos. Os Dados de Código podem também ser criados em uma tentativa de reduzir requisitos de espaço em disco. Os requisitos podem também incluir a habilidade de manter Dados de Código. Todos esses são requisitos não-funcionais do usuário.

Os Dados de Código são uma implementação de requisitos não-funcionais do usuário. Como conseqüência, os Dados de Código podem influenciar o tamanho não-funcional do produto de software, mas *não* o tamanho *funcional* do mesmo ["A Framework for Functional Sizing", IFPUG 2003"].

# Metodologia

O impacto dos dados de código pertencerem à dimensão não-funcional é que nem os Dados de Código, nem as transações que os mantém devem ser contados.

## Introdução

A seção “Identificando Dados de Código” abaixo fornece um processo passo-a-passo para a identificação do que é e do que não é dado de código. A seção é geralmente referenciada no passo “Identificando Arquivos Lógicos”, como definido na parte 3 – capítulo 2, onde os dados de código são desconsiderados. Conforme previamente declarado, os dados de código não são considerados parte do tamanho funcional. Isso tem várias consequências. Para sermos perfeitamente claros neste Guia de Implementação, resumimos as consequências abaixo.

## Consequências

| | |
|---|---|
| 1 | **Não conte Dados de Código como um Arquivo Lógico**<br>Uma consequência de desconsiderar os Dados de Código é que os mesmos não podem ser considerados ALI ou AIE. |
| 2 | **Não conte Dados de Código como um DER ou RLR**<br>Uma consequência de desconsiderar os Dados de Código é que os mesmos não podem ser considerados um RLR ou DER em um ALI ou AIE. |
| 3 | **Não conte Dados de Código como um ALR**<br>Uma consequência de desconsiderar os Dados de Código é que os mesmos não podem ser considerados ALR ao avaliar a complexidade de uma função transacional (EE, SE, CE), porque não se trata de um arquivo lógico. |
| 4 | **Não conte Funções Transacionais de Dados de Código**<br>Uma consequência dos Dados de Código serem uma parte de outra dimensão (a dimensão não-funcional em contraste à dimensão funcional) é que a manutenção de Dados de Código ou funções de relatório não são consideradas ao se medir o tamanho funcional da aplicação. |

# Identificando Dados de Código

Os tipos de Dados de Código resumidos em "O Que São Dados de Código" e "O Que Não São Dados de Código" podem ser usados como uma ajuda prática para determinar se uma entidade é ou não Dados de Código. Alguns critérios podem se sobrepor em parte. Assim que o critério de uma das subseções tiver sido satisfeito, a entidade deverá ser considerada como Dados de Código e não contada.

Os exemplos fornecidos não são uma lista exaustiva e podem não cobrir todos os casos possíveis. Na dúvida, avalie os tipos de entidade dentro do contexto de "Tipos de Entidades de Dado".

## O Que São Dados de Código

### Introdução

Esses são vários tipos diferentes de Dados de Código, os quais se enquadram em três áreas gerais:

- Dados de Substituição fornecem um código e um nome explicativo ou descrição para um atributo de um objeto de negócio (Substituição é uma condição suficiente mas não necessária para ser considerado Dados de Código).
- Dados Constantes ou Estáticos que raramente mudam.
- Dados com Valores Válidos fornecem uma lista de valores disponíveis para um atributo de um ou mais tipos de objetos de negócio.

### Tipos de Dados de Código

| Substituição | Estáticos ou Constantes | Valores Válidos |
|---|---|---|
| Código + Descrição | Uma Ocorrência | Valores Válidos |
| | Dados Estáticos | Faixa de Valores Válidos |
| | Valores Default | |

Quaisquer desses tipos de Dados de Código podem também incluir outros atributos, como data de início e fim de vigência para definir o período de tempo no qual o valor está disponível. Também podem incluir atributos de tipo auditoria tais como data de criação, criado por (id do usuário), data da última atualização, última atualização por (id do usuário). Ainda, uma diversidade de variações é possível (por exemplo, código + descrição resumida / completa). A presença desses atributos adicionais não afeta o processo de categorização, mas os atributos são considerados parte dos Dados de Código.

## Substituição

**Código + Descrição**

Este tipo de Dado de Código contém um código e um nome explicativo ou descrição. Este tipo de Dado de Código pode servir como um meio para tornar mais ágil a entrada de dados para usuários experientes, o nome / descrição explicativo para usuários menos experientes ou para listagens como em relatórios. Este tipo de Dado de Código também pode ser implementado para economizar espaço de armazenamento ou ser um resultado de normalização. Se for dado de substituição, é Dado de Código e não é contado.

**Exemplos**

- Estados: Código do Estado, Nome do Estado
- Cores: Código da Cor, Descrição da Cor

**Variações**

- Código, Idioma, Descrição (Para descrições em múltiplos idiomas)
- Código, Descrição Resumida, Descrição Completa, Abreviatura

## Estáticos ou Constantes

**Uma Ocorrência**

Este tipo de Dado de Código contém uma e apenas uma ocorrência independentemente da quantidade de atributos. Os Dados de Código tem apenas um registro de dados e os atributos são relativamente constantes; podem mudar, mas muito raramente.

**Exemplos**

- Uma entidade com dados sobre uma organização em particular; por exemplo, nome e endereço.
- Software COTS com o nome da companhia aérea, customizado pela organização usuária

**Dados Estáticos**

Este tipo de Dado de Código contém dados que são basicamente estáticos. A quantidade de instâncias de dados estáticos pode mudar, mas muito raramente, e o conteúdo de uma instância raramente muda.

**Exemplos**

- Uma entidade elementos químicos: símbolo, número atômico, descrição
- As tabelas de pontos de função para valorar os tipos de função e os níveis de complexidade

**Valores Default (template)**

Este tipo de Dado de Código contém valores default para (alguns atributos em) novas instâncias de um objeto de negócio.

## Valores Válidos

**Valores Válidos**

Este tipo de Dado de Código fornece uma lista de valores válidos para um atributo de um ou mais tipos de objetos de negócio. Este tipo de Dado de Código é implementado para satisfazer requisites como reduzir erros e aumentar a facilidade de uso pelo usuário. Este tipo de Dado de Código é normalmente usado para listar valores disponíveis para seleção pelo usuário e/ou validar a entrada fornecida pelo mesmo. Este tipo de Dado de Código contém dados basicamente estáticos; se não forem, podem ser Dados de Referência ou Dados de Negócio.

**Exemplos**

- Nome do estado: Contém todos os valores válidos para o atributo nome do estado
- Código do estado: Contém todos os valores válidos para o atributo código do estado
- Cor: Contém todos os valores válidos para o atributo cor de um objeto de negócio

**Faixa de Valores Válidos**

Este tipo de Dado de Código contém dados basicamente estáticos; se não forem podem ser Dados de Referência.

**Exemplos**

- Faixa de Números de Telefone Permissíveis: menor número de telefone, maior número de telefone.
- Faixa de temperatura térmica.

## O Que Não São Dados de Código

Esta seção descreve dados que não são considerados Dados de Código porque são Dados de Negócio ou Dados de Referência. A Parte 1 e a Parte 2 do CPM – Capítulo 2 (Arquivos Lógicos) contém as regras para esses tipos de entidades. Algumas vezes tabelas são chamadas ‘tabelas de código’ mas são na realidade Dados de Referência ou mesmo Dados de Negócio.

### Exemplos

Exemplos de Dados de Negócio ou Dados de Referência que não devem ser considerados Dados de Código:

- Tipos de entidade com montantes financeiros, taxas de câmbio, e alíquotas de impostos, se não forem constantes. Esses dados não restringem valores válidos; ao invés disso, adicionam significado para um valor dentro de uma faixa em particular.
- Dados de controle: dados mantidos pelo usuário que contém regras de negócio dizendo à aplicação o que fazer ou como se comportar.
- Tabela de Taxa de Câmbio: Taxa de Câmbio contém a taxa de câmbio da moeda corrente do país na conversão para dólar. Não é possível substituir o código pela taxa de câmbio do país, o dado é essencialmente não estático, e os dados apoiam as atividades de negócio; portanto, isto é um exemplo de Dados de Referência.
- Faixa de percentual de tributação para um Sistema de Percentuais Progressivos: um percentual de tributação diferente é aplicável para diferentes faixas de receita. Contém um valor mínimo e máximo para cada percentual de tributação. Entretanto, não se pode substituir o valor pela entidade, e os dados apoiam atividades do negócio. O percentual de tributação não restringe a receita. Portanto, este é um exemplo de Dados de Referência.

# Considerando Dados de Código e Transações de Dados de Código

Os Dados de Código como identificados em “O Que São Dados de Código” representam a implementação de requisitos não-funcionais do usuário ao invés da implementação de Requisitos Funcionais do Usuário. Consequentemente, os Dados de Código e as transações que os mantém não contam para o tamanho funcional da aplicação.

Contudo, os dados que são parte de requisitos não-funcionais podem ser medidos usando uma medida distinta para dimensionar o tamanho não-funcional.

Neste momento não existe um método específico para contar Dados de Código, bem como suas funções de manutenção e relatório, a fim de produzir um tamanho para a dimensão não-funcional.

.

# Bibliografia

As seguintes fontes foram consultadas ou citadas neste capítulo:

Definitions and Counting Guidelines for the Application of Function Point Analysis: A Practical Manual, Version 2.2. (NESMA, 2003).

ISBN: 90-76258-17-1.

Nota: Esse manual é também chamado Manual de Práticas de Contagem da NESMA. Descreve a metodologia padrão de APF e muitos aspectos relacionados à aplicação da mesma. Pode ser usado em conjunto com o manual do IFPUG. Para mais informações, visite o web site da NESMA www.nesma.org.

IFPUG "A Framework for Functional Sizing", IFPUG, 2003.

ISO/IEC 14143-1:2007 Information technology – Software measurement Functional size measurement – Part 1: definition of concepts, ISO/IEC, 2007

Esta página foi intencionalmente deixada em branco.

# Parte 3 Capítulo 2

# Arquivos Lógicos

**Introdução**

Este capítulo aplica as definições e regras referentes aos arquivos lógicos, usando um processo descritivo para a identificação e classificação das funções de dados.

Essas orientações ilustram a identificação de funções de dados a partir de modelos de dados normalizados. Uma visão geral de normalização de dados é fornecida para apoiar essa abordagem. Contudo, isso não exclui o uso dessas orientações em ambientes onde técnicas alternativas para a modelagem de dados ou objetos forem empregadas.

**Conteúdo**

Este capítulo inclui as seguintes seções:

# Resumo da Metodologia

## Introdução

Um processo passo-a-passo para estabelecer um conjunto de arquivos lógicos (Arquivo Lógico Interno e Arquivo de Interface Externo) é usado, onde cada passo enxerga os dados em um nível de detalhe mais refinado. Os detalhes de cada passo são explicados nas seções subsequentes.

| Passo | Ação |
|---|---|
| 1 | **Identificar Arquivos Lógicos**<br>Para cada entidade lógica de dados, identifique como as entidades relacionadas são agrupadas em arquivos lógicos, os quais refletem a "visão do usuário". Por exemplo, determine se as entidades de dados são um arquivo lógico independente por si só ou se entidades relacionadas devem ser agrupadas em um único arquivo lógico. Este passo é explicado na Seção "Passo 1: Identificar Arquivos Lógicos". |
| 2 | **Classificar Arquivos Lógicos**<br>Cada arquivo lógico identificado é classificado como ALI ou AIE. Este passo é explicado na Seção "Passo 2: Classificar Arquivos Lógicos". |
| 3 | **Identificar Tipos de Dados Elementares**<br>Para cada arquivo lógico, identifique os elementos de dados usados pela aplicação medida. Este passo é explicado na Seção "Passo 3: Identifique Tipos de Dados Elementares" |
| 4 | **Identificar Tipos de Registro Elementares**<br>Para cada arquivo lógico, identifique como os dados relacionados são agrupados em tipos de registro elementares, os quais refletem a "visão do usuário". Este passo é explicado na Seção "Passo 4: Identifique Tipos de Registro Elementares". |

O passo com o maior impacto no tamanho funcional é o Passo 1: Identificar Arquivos Lógicos, porque a identificação da quantidade correta de arquivos lógicos é crucial para alcançar a consistência entre contagens. Os passos restantes influenciam o tamanho funcional em um grau consideravelmente menor, porque não afetam a quantidade de arquivos lógicos, mas apenas o seu tipo e classificação de complexidade.

# Conceitos de Modelagem de Dados

## Introdução

Uma revisão das definições usadas no domínio da análise de dados (o qual inclui modelagem lógica e física de dados) pode fornecer uma base para o entendimento, assim como esclarecer o propósito das regras de práticas de contagem, conforme relacionadas à identificação de Arquivos Lógicos, Tipos de Registro Elementares e Tipos de Dados Elementares. Um profundo entendimento dos conceitos de modelagem de dados está implícito na Análise de Pontos de Função, ao medir Funções de Dados da maneira apropriada e correta. A seção “Termos da Modelagem de Dados” resume os termos da modelagem de dados.

## Conceitos Chave em Modelagem de Dados

Uma revisão das definições usadas no domínio de análise de dados (o qual inclui modelagem de dados e SGBD) pode fornecer uma base para o entendimento, assim como esclarecer o propósito das regras de práticas de contagem na medida em que se relacionam especificamente a Arquivos Lógicos, Tipos de Registro Elementares (RLRs) e Tipos de Dado Elementares (DERs). Um entendimento de conceitos de dados é assumido na aplicação das orientações descritas no próximo capítulo para medir as Funções de Dados de maneira apropriada e correta.

## Tipo de Entidade

**Definições de Entidade (ou Tipo de Entidade)**

- Qualquer pessoa, local, coisa, evento ou conceito distinto sobre o qual informação é mantida (Thomas Bruce, 1992)
- Uma coisa que pode ser identificada de forma distinta (Peter Chen, 1976).
- Qualquer objeto distinto representado em uma base de dados (C.J. Date, 1986)
- Uma entidade de dados representa alguma "coisa" que será armazenada para referência futura. O termo entidade se refere à representação lógica dos dados (Clive Finkelstein, 1989)
- A palavra entidade significa qualquer coisa sobre a qual armazenamos informação (por exemplo, um cliente, fornecedor, ferramenta mecânica, empregado, poste, assento de companhia aérea, etc.). Para cada entidade, certos atributos são armazenados (James Martin, 1989)
- Uma entidade também pode representar o relacionamento entre duas ou mais entidades, chamadas entidades associativas (Michael Reingruber, 1994)
- Uma entidade pode representar um subconjunto de informações relevante para uma instância de uma entidade, chamada entidade subtipo (também conhecido como entidade secundária ou entidade categoria) (Michael Reingruber, 1994)

Para resumir, uma entidade:

- é um objeto de dados principal sobre o qual informações são coletadas
- é uma pessoa, local, coisa ou evento de informação
- pode ter uma instância (uma ocorrência)
- é uma coisa fundamental relevante para o usuário, sobre a qual uma coleção de fatos é mantida; uma associação entre entidades que contém atributos é por si só uma entidade
- envolve informações, uma representação de coisas similares que compartilham características ou propriedades
- é frequentemente representada graficamente em um modelo de dados por um retângulo, com o nome escrito dentro do mesmo

### Elemento de Dados

No mundo da modelagem de dados, o elemento básico é chamado ***elemento de dados*** ou ***item de dados***. Ele é

- o componente fundamental
- a partícula fundamental no universo do sistema de informações (Gary Schutt)
- a menor unidade de dados com nome que tem significado no mundo real/do usuário (Graeme Simsion)

### Atividades da Modelagem de Dados

A modelagem de dados aborda itens de dados, registros lógicos e arquivos. Um **Sistema de Arquivos** é composto de registros de itens de dados. **Itens de dados** são definidos como a menor unidade de dados com nome que tem significado para o mundo real. Um grupo de itens relacionados tratado como uma unidade é conhecido como um **registro**. Um **arquivo** é uma coleção de registros de um único tipo.

Na implementação física de dados por meio de bases de dados relacionais, são usados os seguintes termos: um item de dados é chamado “atributo” ou “coluna”, um registro é chamado “linha” ou “tupla” e um arquivo é chamado “tabela”. Esses termos não mudam o significado básico dos conceitos.

**Mapeando Conceitos de Dados para a Terminologia de Pontos de Função**

Podemos ainda mapear esses termos para a análise de pontos de função, conforme mostrado na seguinte matriz:

| Conceito da Modelagem de Dados | Termo da Modelagem de Dados | Termo de Base de Dados Relacional | Termo da APF | Conceito da APF |
|---|---|---|---|---|
| Menor unidade de dado com nome que tem significado para o mundo real | Item de Dados | Atributo ou Coluna | Tipo de Dado Elementar (DER) | Um tipo de dado elementar (DER) é um campo único, não-repetido, reconhecido pelo usuário |
| Grupos de itens relacionados os quais são tratados como uma unidade | Registro | Linha ou Tupla | Tipo de Registro Elementar (RLR) | Um tipo de registro elementar (RLR) é um subgrupo de elementos de dados reconhecido pelo usuário e armazenado em um ALI ou AIE |
| Coleção de registros de um único tipo | Arquivo | Tabela | Arquivo Lógico (Arquivo Lógico Interno - ALI ou Arquivo de Interface Externa - AIE) | Arquivo refere-se a grupos de dados logicamente relacionados e não à implementação física desses grupos de dados |

Uma vez que todos os dados tenham sido identificados, o analista de dados aplica várias regras de normalização para representar graficamente os dados em vários Diagramas de Entidade-Relacionamento. Um resumo das regras de normalização pode ser encontrado na Seção “Termos da Modelagem de Dados”.

## Termos da Modelagem de Dados

**Entidade (ou Tipo de Entidade)**

- Principais objetos de dados sobre os quais informações são coletadas
- Pessoa, local, coisa ou evento de informação
- Instância de entidade (uma ocorrência)
- Representada graficamente por um retângulo, com o nome da entidade escrito em seu interior
- Uma coisa fundamental relevante para o usuário, sobre a qual uma coleção de fatos é mantida. Uma associação entre entidades que contém atributos é por si só uma entidade

**Tipo de Entidade Associativa**

Um tipo de entidade que contém atributos que descrevem em mais detalhe um relacionamento de muitos para muitos entre dois outros tipos de entidades.

**Tipo de Entidade Atributiva**

Um tipo de entidade que descreve em mais detalhe uma ou mais características de outro tipo de entidade.

**Entidade Subtipo**

Uma subdivisão de tipo de entidade. Um subtipo herda todos os atributos e relacionamentos de seu tipo de entidade pai e pode ter atributos e relacionamentos adicionais próprios.

**Relaciona-mentos**

Representam associações do mundo real entre uma ou mais entidades

- Um-para-Um
- Um-para-Vários
- Vários-para-Vários
- Representados por uma linha a qual conecta as entidades.
- O nome do relacionamento é escrito ao lado da linha

Relacionamentos são definidos por como as entidades são conectadas:

- Opcionais, apresentadas no texto com parêntesis 1:(N), (1):(N)
- Obrigatórias, apresentadas no texto sem parêntesis 1:1, 1:N

**Atributos**

- Uma característica de uma entidade. Atributos são geralmente análogos a Tipos de Dados Elementares (DERs).

**Normalização** Dados são normalizados pelo uso de 5 regras

1. Elimine Grupos Repetidos (1ª Forma Normal)
2. Elimine Dados Redundantes (2ª Forma Normal)
3. Elimine Colunas não dependentes da Chave (3ª Forma Normal)
4. Isole Relacionamentos Independentes múltiplos (nenhuma tabela pode conter dois ou mais relacionamentos 1:N ou N:M – 4ª Forma Normal)
5. Isole relacionamentos múltimos semanticamente relacionados (restrições práticas podem justificar a separação de relacionamentos logicamente relacionados de muitos para muitos – 5ª Forma Normal)

Ao realizar a análise de pontos de função, é preferível analisar o modelo lógico em 3 ª Forma Normal.

Ignore múltiplas entidades incluídas em função da tecnologia (em geral 5ª forma normal)

## Conceitos de Entidade-Relacionamento

Uma vez que todos os dados necessários tenham sido identificados, o analista de dados aplica várias regras de normalização para representar graficamente os dados em vários Diagramas de Entidade-Relacionamento. A tabela a seguir pode ser aplicada para melhor entender o conceito de Tipo de Registro Elementar.

| **Conceito de Entidade-Relacionamento** | **Termo E-R** | **Termo na APF** | **Definição do IFPUG** |
|---|---|---|---|
| Objeto de dados principal sobre o qual informações são coletadas (pessoa, local, coisa ou evento); um item de relevância fundamental para o usuário sobre o qual uma coleção de fatos é mantida | Entidade ou Tipo de Entidade | Arquivo Lógico Interno (ALI) ou Arquivo de Interface Externa (AIE) | O arquivo se refere a um grupo de dados logicamente relacionados e não à implementação física desses grupos de dados |
| Um tipo de entidade que contem atributos que descrevem complementarmente relacionamentos entre outras entidades | Tipo de Entidade Associativa | Tipo de Registro Elementar (RLR) | Subgrupo de elementos de dados reconhecido pelo usuário em um ALI ou AIE (opcional ou obrigatório) |
| Um tipo de entidade que descreve complementarmente uma ou mais características de outro tipo de entidade | Tipo de Entidade Atributiva | Tipo de Registro Elementar (RLR) | Subgrupo de elementos de dados reconhecido pelo usuário em um ALI ou AIE (opcional ou obrigatório) |
| Uma divisão de um tipo de entidade, a qual herda todos os atributos e relacionamentos de seu tipo de entidade pai; pode ter atributos e relacionamentos adicionais, únicos | Entidade Subtipo | Tipo de Registro Elementar (RLR) | Subgrupo de elementos de dados reconhecido pelo usuário em um ALI ou AIE (opcional ou obrigatório) |

# Passo 1: Identificar Arquivos Lógicos

## Background

Na APF, um arquivo lógico é um grupo de dados conforme visto pelo usuário. Um arquivo lógico é composto de uma ou mais entidades de dados. Este capítulo fornece orientações sobre como agrupar as entidades candidatas identificadas em um ou mais arquivos lógicos.

O processo consiste dos seguintes passos, todos os quais são explicados em detalhe nos seguintes parágrafos desta seção:

### Passo 1

| Subpasso |
|---|
| 1. Identifique todos os dados ou informações de controle reconhecidos pelo usuário logicamente relacionados no escopo da contagem |
| 2. Exclua as entidades não mantidas por qualquer aplicação. |
| 3. Agrupe em arquivos lógicos as entidades relacionadas que forem entidades dependentes |
| 4. Exclua aquelas entidades referenciadas como dados de código |
| 5. Exclua as entidades que não contenham atributos exigidos pelo usuário |
| 6. Remova as entidades associativas que contenham atributos adicionais não exigidos pelo usuário e entidades associativas que contenham apenas chaves estrangeiras; agrupe os atributos chave estrangeira com as entidades principais. |

O passo mais difícil é o agrupamento de dados (subpasso 3). O agrupamento final de dados em arquivos lógicos é o resultado do efeito combinado de dois métodos de agrupamento:

- Método a) é orientado pelo processo, baseado nas transações de usuário na aplicação
- Método b) é orientado pelos dados, baseado nas regras de negócio

Contudo, como as transações do usuário também são (ou deveriam ser) baseadas em regras de negócio, cada método apoia o outro. Essa abordagem dupla pode revelar eventuais deficiências na especificação funcional e torna o processo de identificação de arquivos lógicos confiável e passível de repetição.

### Subpasso 1.1 Identificar dados ou informações de controle logicamente relacionados reconhecidos pelo usuário dentro do escopo da contagem

Antes de tomar a decisão sobre quais entidades devem ser agrupadas em arquivos lógicos como um conjunto, deve-se determinar quais entidades candidatas devem ser consideradas para o agrupamento lógico das entidades (subpasso 1.3) e quais devem ser excluídas. Os passos a seguir ajudarão a identificar essas entidades de uma maneira passível de repetição.

Os princípios de orientação geral da Parte I – Medir Funções de Dados são claros: considere apenas entidades significativas e requeridas pelo usuário.

Preste atenção especial quando da identificação de arquivos lógicos a partir de um modelo (normalizado) de dados:

- Não assuma que todas as entidades são arquivos lógicos; por exemplo, arquivos de índice, entidades em um modelo de dados físico.
- Arquivos lógicos podem existir em uma perspectiva do usuário, mas em alguns casos podem não ser identificados no modelo (normalizado) de dados; por exemplo, arquivos históricos contendo dados agregados. Não esqueça de incluir esses arquivos lógicos no restante do processo.

### Subpasso 1.2 Exclua entidades não mantidas por qualquer aplicação

Determine quais entidades não são mantidas por um processo elementar nesta ou em outra aplicação. Exclua essas entidades de considerações usubsequentes, pois as mesmas não são contadas.

### Subpasso 1.3 Agrupe em arquivos lógicos as entidades relacionadas que são entidades dependentes

Para cada entidade de dados restante, identifique como as entidades relacionadas devem ser agrupadas em arquivos lógicos, os quais refletem a "visão do usuário"; isto é, determine se as entidades de dados constituem por si mesmas um arquivo lógico independente ou se as entidades relacionadas devem ser agrupadas em um único arquivo lógico.

Identifique a visão do usuário (= visão de negócio) do agrupamento de dados investigando:

a) Como os dados são acessados como um grupo por processos elementares dentro da fronteira da aplicação (Subpasso 1.3a, página 2-14)

b) Os relacionamentos entre as entidades e a sua interdependência baseada nas regras de negócio (Subpasso1.3b, da página 2-15 até a 2-24).

### Subpasso 1.4 Excluir entidades referenciadas como dados de código

Filtrar dados de código. Dados de código são incluídos como resposta a um requisito não-funcional do usuário (requisitos de qualidade, implementação física e/ou razão técnica). Dados de código são explicados em detalhe na Parte 3 – Capítulo 1 "Dados de Código".

### Subpasso 1.5 Excluir entidades que não contenham atributos requeridos pelo usuário

Determinar quais entidades não contém atributos reconhecidos e requeridos pelo usuário, contendo apenas atributos não funcionais. Exemplos de atributos não funcionais são aqueles que existem como um resultado de um projeto ou consideração de implementação; por exexplo, índice de arquivos criados por razões de performance, tais como índices alternados (ver Parte 1 – Medir Funções de Dados). Excluir tais entidades de considerações posteriores; as mesmas não são contadas como um arquivo lógico ou RLR.

### Subpasso 1.6 Remover entidades associativas que contenham atributos adicionais não requeridos pelo usuário e entidades associativas que contenham apenas chaves estrangeiras; agrupar atributos de chaves estrangeiras com as entidades primárias

**1.6.1** Determinar quais entidades são entidades associativas. Uma entidade associativa contém chaves estrangeiras de entidades conectadas juntamente com outros atributos. Note que duas situações podem surgir como resultado:

a) Os atributos adicionais não-chave são um resultado do projeto ou consideração de implementação, ou existem para satisfazer um requisito técnico (não requeridos pelo usuário; ex.: um campo de data/hora com o propósito de recuperação de dados). Estes atributos técnicos não são contados como elementos de dados. Trate estas entidades como entidades key-to-key (veja abaixo).

b) Os atributos adicionais não-chave são necessários para satisfazer os requisitos funcionais do usuário e são requeridos pelo usuário. Estas entidades são avaliadas nas próximas sessões: Identificar Arquivos Lógicos (Passos 1.3a/1.3b).

**Exemplo:**


*Timestamp é normalmente um atributo técnico não reconhecido pelo usuário. Neste caso, para a entidade Itens do Pedido, aplica-se a situação 1.3a. A entidade key-to-key é resolvida pela inclusão do Código do Produto como uma chave estrangeira no Pedido e do Código do Pedidono Produto (passo 1.6).*

**1.6.2** Determinar quais entidades são entidades key-to-key (intersecção); por ex., eles possuem apenas chaves como elementos de dados e não têm nenhum outro atributo não-chave.

Estas entidades normalmente representam a implementação de uma relação muitos para muitos (N:M) em um modelo de dados normalizado. Existem apenas por razões de modelagem de dados e projeto de banco de dados e não como resultado de um requisito do usuário.

Exclua estas entidades de outras considerações; elas não são contadas como arquivo lógico ou RLR. De acordo com as regras (Parte 1), o atributo que faz referência (chave estrangeira) é contado como um elemento de dado em ambas entidades conectadas pela entidade key-to-key. Veja também as diretrizes na Sessão “Passo 3: Identificar Tipos de Elementos de Dados”.

**Verificação Final**

Verificar se todas as entidades restantes são resultado de requisitos funcionais do usuário. Estas entidades e as relações e interdependências entre as mesmas serão abordados na próxima sessão: Identificar Arquivos Lógicos (subpasso 1.3a/1.3b).

## Identificar Arquivos Lógicos Utilizando o Método de Processos Elementares (Subpasso 1.3a)

A visão de negócio do usuário sobre os dados é refletida em como as transações do usuário acessam os dados.

Reveja como os processos elementares dentro da fronteira da aplicação mantêm as entidades. Se várias entidades são sempre *criadas* juntas e *excluídas* juntas então esta é uma forte indicação de que as mesmas devem ser agrupadas dentro de um único arquivo lógico. Reveja também os processos elementares usados para extrair os dados, para determinar se o processo de extração acessa o mesmo grupo de entidades. Nota: as transações que modificam dados frequentemente têm como alvo apenas uma entidade no grupo; dessa forma, as transações de modificação não fornecem uma orientação tão eficaz para agrupamento de dados quanto as transações de inclusão e exclusão.

**Exemplo**

Um pedido de compra do cliente é um grupo único de dados a partir da perspectiva do negócio do usuário; ele é composto dos Dados Básicos do Pedido (cliente, endereço, data, etc.) e dos detalhes sobre cada item pedido. A partir da perspectiva do negócio, um pedido não pode ser criado sem pelo menos um item e se o pedido for excluído, tanto os dados básicos como todos os seus itens serão excluídos. Entretanto, os dados básicos e os itens podem ter transações de manutenção independentes; por ex., a alteração do status do pedido é uma função diferente da alteração dos itens do pedido. As funções de *inclusão* e *exclusão* indicam, a partir da perspectiva do usuário, que "pedido" é um arquivo lógico único que agrupa os dados básicos do pedido e os itens do pedido.

Utilize o subpasso 1.3a para validar os grupos de dados lógicos candidatos que foram identificados.

# Identificar Arquivos Lógicos Utilizando o Método de (In)Dependência de Entidades (Subpasso 1.3b)

## Introdução

O Método de (In)Dependência de Entidades, como definido e explicado nesta seção, fornece um método reproduzível para identificar corretamente Arquivos Lógicos (ALs) a partir de um modelo de dados. Nesta seção, o termo "entidade" refere-se a uma entidade em um modelo de dados normalizado (normalmente na terceira forma normal).

A Seção "Tipos de Relacionamentos" explica os diferentes tipos de relacionamentos e as diferenças entre os conceitos "relacionamento obrigatório/opcional" e "entidades dependentes /independentes ".

A Seção "(In)Dependência de Entidades Ilustrada para Todos os Tipos de Relacionamentos" explica o método em mais detalhes para cada tipo de relacionamento.

A Seção "Resumo: de Entidades para Arquivos Lógicos via (In)Dependência de Entidades" resume os tipos de relacionamentos e as condições para quando contar um AL.

O Método de Dependência de Entidades agrupa entidades pela avaliação dos relacionamentos e interdependências das entidades em comparação com as regras de negócio. Os princípios do guia são *entidades independentes* e *entidades dependentes*.

**Entidades Independentes**

*Entidade independente* significa uma entidade que é significativa ou tem sentido para o negócio por si só, sem a presença de outras entidades.

**Entidades Dependentes**

*Entidade dependente* significa uma entidade que não é significativa ou não tem sentido para o negócio por si só, sem a presença de outras entidades, de modo que:

- uma ocorrência da entidade X deve estar ligada a uma ocorrência da entidade Y
- a eliminação de uma ocorrência da entidade Y resulta na eliminação de todas as ocorrências relacionadas da entidade X

**Nota**

Não confunda o conceito de *entidade independente/entidade dependente* com o conceito de *relacionamento opcional/obrigatório*. Os exemplos na Seção "(In)Dependência de Entidade Ilustrada para Todos os Tipos de Relacionamentos" mostram claramente que estes são conceitos diferentes.

**Determinar Dependência**

Para determinar se a entidade B é dependente ou independente da entidade A, é preciso determinar:

*"B é significativa para o negócio independentemente da ocorrência de A ligada a ela?"*

Um teste simples para determinar a situação (entidade dependente ou independente) é o seguinte. Mesmo que não haja requisito do usuário para a exclusão, (ainda assim) faça a pergunta:

*"Suponha que nós quiséssemos excluir uma ocorrência "a" da entidade A; o que aconteceria à ocorrência "b" da entidade B ligada a "a"?"*

Dependendo das regras do negócio, distinguimos duas situações essencialmente diferentes:

**Situação 1**

Se, de acordo com as regras de negócio, uma ocorrência de B não tem significado/importância independente para o usuário e pode também ser excluída, então aparentemente a ocorrência de B não tem significado para o usuário independentemente da ocorrência correspondente de A. A entidade B é considerada uma entidade dependente de A. As entidades A e B devem ser agrupadas juntas no mesmo arquivo lógico.

**Situação 2**

Se a ocorrência de B tem significado para o negócio mesmo independentemente da ocorrência correspondente de A, as regras de negócio não permitirão a exclusão da ocorrência de B. As entidades A e B serão consideradas arquivos lógicos separados.

Avaliar o modelo de dados de um sistema de informação por meio da avaliação de todos os pares de entidades ligadas resulta na identificação dos arquivos lógicos.

Na Seção "Entidades (In)Dependentes Ilustrada para Todos os Tipos de Relacionamento" este método é explicado em mais detalhes para diferentes tipos de relacionamentos.

A Seção "Resumo: de Entidades para Arquivos Lógicos via (In-) Dependência de Entidades " resume como contar cada tipo de relacionamento na APF.

## Tipos de Relacionamentos

Antes de assumir como conclusivos os princípios da (In)Dependência de Entidades para todos os tipos de relacionamento, deve-se entender claramente os diferentes tipos/naturezas dos relacionamentos. Esta seção explica os diferentes tipos, assim como os conceitos de “opcional” e “obrigatório”.

**Exemplo**

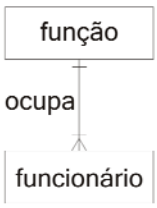
Duas entidades, Função e Funcionário, por exemplo, podem ser conectadas entre si via um relacionamento; por ex.: “ocupa”.

**Natureza do Relacionamento**

A natureza do relacionamento determina quantos funcionários podem trabalhar em uma função de acordo com o modelo de dados (0, 1 ou mais) e em quantas funções um funcionário pode trabalhar (0, 1 ou mais).

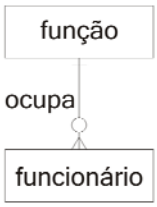
**1 : N**

Assuma que as regras de negócio determinem que vários funcionários (no mínimo 1) podem ser utilizados em uma função, e que um funcionário tem que trabalhar em uma (e apenas uma) função. Neste caso dizemos que o relacionamento entre Função e Funcionário é 1:N



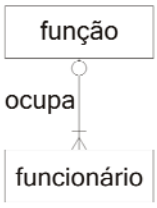

**1 : (N)**

É mais provável que as regras de negócio determinem que uma função *pode* estar vaga, isto é, nenhum funcionário tenha sido alocado para a função. Neste caso o relacionamento entre Função e Funcionário é opcional e definido como 1:(N).

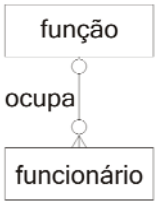



**(1) : N**

Se as regras de negócio determinam que um funcionário pode existir sem uma função, mas uma função sempre tem um funcionário a ela alocado, definimos o relacionamento entre Função e Funcionário como (1):N



**(1) : (N)** Na situação onde uma função pode estar vaga e um funcionário pode existir sem uma função, ambos os lados do relacionamento são opcionais. O relacionamento entre Função e Funcionário é definido como (1):(N).




**Conceito "Obrigatório/Opcional" versus "(In)Dependência de Entidades".**

Para deixar clara a diferença entre os conceitos de "relacionamento obrigatório/opcional" e "dependência/independência de entidades", assuma, como um exemplo, a seguinte extensão das regras de negócio "funcionário(s) não é(são) permitido(s) sem uma função" (relacionamento do tipo 1:N e 1:(N)).

Quando uma função se torna obsoleta, isto não significa que os funcionários não são mais significativos para o negócio. Um funcionário tem significado para o negócio independente da função relacionada. Funcionário é uma entidade independente de função. Devido ao relacionamento obrigatório com função, todos os funcionários têm que ser alocados a uma nova função, antes de a função poder ser excluída.

Então pode acontecer que uma ocorrência da entidade B (ex. Funcionário) possa ter um link obrigatório com uma ocorrência da entidade A (por ex. Função) no relacionamento A:B entre as entidades A e B, mas aquela entidade B é por si só significativa para o negócio. Neste caso, quando alguém quiser excluir uma ocorrência da entidade A, tem-se que antes reatribuir uma ocorrência ligada de B para outra ocorrência de A.

## (In)Dependência de Entidade Ilustrada para Todos os Tipos de Relacionamento

**(In)Dependência de Entidade em um Relacionamento (1):(N)**

**(1) : (N)** Se um relacionamento entre duas entidades A e B é bilateralmente opcional, as entidades podem existir independentemente e (todas ocorrências de) A e B são significativas para o negócio independentemente da(s) ocorrência(s) relacionada(s) com a outra entidade.

Então, A e B são consideradas entidades independentes uma da outra. A APF conta as entidades A e B como dois arquivos lógicos separados, conforme indicado na tabela da Seção "Resumo: de Entidades para Arquivos Lógicos via (In-)Dependência de Entidades".

**(In)Dependência de Entidade em um Relacionamento 1:(N)**

**1 : (N)** Em um relacionamento 1:(N) entre duas entidades A e B (veja figura 1), pode existir uma ocorrência da entidade A para nenhuma, uma ou muitas ocorrências da entidade B relacionadas. Por outro lado, cada ocorrência de B tem que ser associada a uma ocorrência de A.

**Exemplo** No relacionamento 1:(N) entre Funcionário e Filho (ou Dependente) em uma Aplicação de RH, um Funcionário deve ter 0, 1 ou muitos Dependentes a ele relacionados, mas um Dependente tem que estar relacionado a um (e apenas um) Funcionário (veja figura 2).

Como B deve ser relacionado a um A, isto levanta a questão se B é dependente ou independente de A.

Para determinar se a entidade B é dependente ou independente de A, é necessário responder:

*"B é significativo para o negócio independentemente do A a ela relacionado?"*

Veja um teste simples para diferenciar a dependência e independência de entidades.

Mesmo que não existam requisitos do usuário para exclusão, faça a seguinte pergunta:

*"Suponha que desejamos excluir uma ocorrência da entidade A; o que acontecerá com as ocorrências relacionadas da entidade B que têm um relacionamento obrigatório com uma ocorrência de A?"*

As regras de negócio podem resultar em duas possibilidades:

**Situação 1** Se a exclusão de A for permitida, todas as ocorrências de B relacionadas também deverão ser excluídas, pois o negócio não está mais interessado nas ocorrências de B. Por exemplo (veja figura 2): Uma aplicação de RH mantém informações sobre funcionários e seus dependentes. Assuma que as regras de negócio definiram que quando um Funcionário (A) deixa a companhia, não tem mais sentido para o negócio manter a informação sobre os dependentes (B).

**Situação 2**

A exclusão de A não é permitida enquanto ocorrências de B ainda estiverem a ela relacionadas, pois o negócio está ainda interessado nas ocorrências de B, mesmo além do contexto do A correspondente. Por exemplo (veja figura 3): Uma organização adota crianças e designa cada criança a um funcionário. O funcionário é a pessoa de contato entre a companhia e a criança. No caso de um funcionário deixar a companhia, as informações sobre a criança associada (do funcionário desligado) ainda são significativas para o negócio. Então, antes que se permita a exclusão do Funcionário (A), tem-se que primeiramente atribuir a Criança associada (B) a outro Funcionário (A) (pois a natureza deste relacionamento não permite uma Criança sem um relacionamento com Funcionário).

Na situação (1) dizemos que B é uma *entidade dependente* de A, e na situação (2) que B é uma *entidade independente* de A.

A APF conta as entidades A e B como um único arquivo lógico na situação (1) (*dependência*), enquanto na situação (2) A e B são arquivos lógicos separados (*independência*) conforme indicado na tabela da Seção "Resumo: de Entidades para Arquivos Lógicos via (In-)Dependência de Entidades".

Ilustração do relacionamento 1:(N):

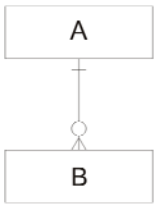



Fig. 1:

Cada entidade do tipo A pode referenciar 0, 1 ou muitas entidades do tipo B. Uma entidade do tipo B tem que referenciar exatamente uma entidade do tipo A.

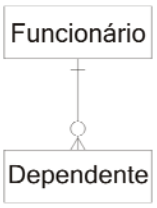



Fig. 2:

A aplicação de RH mantém informações sobre funcionários e seus dependentes.

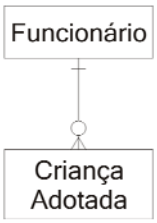



Fig. 3:
A aplicação de RH mantém informações sobre Funcionários e sobre as Crianças Adotadas que são designadas para um Funcionário.

As figuras 2 e 3 possuem modelos de dados similares, mas diferentes regras de negócio resultam em diferentes arquivos lógicos identificados.

**(In)Dependência de Entidade em um Relacionamento (1):N**

**(1) : N**

Um relacionamento (1):N entre duas entidades A e B (veja figura 4) pode ser tratado de forma similar. Estes tipos de relacionamentos, entretanto, raramente aparecem na prática.

Em um relacionamento (1):N entre duas entidades A e B, cada A deve ser atribuído a 1 ou muitos Bs. Por outro lado, um B pode (mas não necessariamente) ser atribuído a uma ocorrência de A.

**Exemplo**

Em um relacionamento (1):N entre Comitê e Membro da Organização, um Comitê tem que ter membros (pelo menos 1). Um membro da organização pode (mas não necessariamente) servir em um Comitê (veja figuras 5 e 6).

Devido a uma ocorrência de A ter que estar relacionada a uma de B, levanta-se a questão se A é dependente ou independente de B.

Para determinar se a entidade A é dependente ou independente de B, precisa-se responder:

*"A é significativa para o negócio independentemente da entidade B a ela relacionada?"*

Veja a seguir um teste simples para diferenciar a dependência e independência de entidades.

Mesmo que não existam requisitos do usuário para exclusão, faça a pergunta:

*"Assuma que temos uma ocorrência da entidade A à qual estão relacionadas uma ou mais ocorrências da entidade B. Suponha que desejamos excluir a última ocorrência relacionada à entidade B; o que aconteceria com esta ocorrência de A, que possui um relacionamento obrigatório com pelo menos uma ocorrência de B?"*

As regras de negócio podem resultar em duas possibilidades:

**Situação 1**

Quando o último B é excluído, o A relacionado é também excluído pois o negócio não se interessa mais por ele. Por exemplo (veja a figura 5): Uma organização tem comitês aos quais membros são atribuídos.

A regra de negócio é que um comitê *deve* ter membros, mas nem todos os membros precisam participar de um comitê. Uma regra de negócio adicional é que a organização encerra um comitê assim que não haja mais membros participando do mesmo; pode-se dizer que os comitês são vistos como grupos de trabalho "ad hoc".

Neste caso quando o último membro de um comitê sai do comitê, *não* tem sentido para o negócio manter informações sobre o comitê. Os dados do comitê são excluídos assim que o último membro deixa o comitê.

**Situação 2**

A exclusão do último B *não* é possível enquanto exista algum A ainda referenciado por ele, pois o negócio está ainda interessado neste específico A, mesmo além do contexto dos Bs que o referenciam. Por exemplo (veja figura 6), uma organização tem comitês aos quais membros são atribuídos.

A regra de negócio é que um comitê *deve* ter membros, mas nem todos os membros precisam participar de um comitê. Comitês são vistos como parte da estrutura organizacional. Eles têm significado para o negócio além dos membros que os servem.

Antes que o último membro de um específico comitê deixe o comitê, um novo membro tem que ser atribuído àquele comitê pois a natureza do relacionamento não permite um comitê sem membros.

Na situação (1), A é aparentemente *não* significativo para o negócio a menos que ele esteja relacionado a um ou mais Bs, enquanto na situação (2) ele *é* significativo.

Na situação (1) nós dizemos que A é uma *entidade dependente* de B e na situação (2) que A é uma *entidade independente* de B .

A APF conta as entidades A e B como um único arquivo lógico na situação (1) (*dependência*), enquanto na situação (2) A e B são arquivos lógicos separados (*independência*), como indicado na tabela da Seção "Resumo: de Entidades para Arquivos Lógicos via (In-)Dependência de Entidades".

Ilustração do relacionamento (1):N:

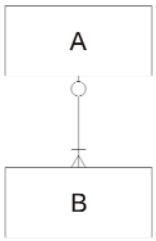



Fig. 4:
Cada entidade do tipo A tem que ser referenciada por 1 ou mais entidades do tipo B; uma entidade do tipo B pode, mas não necessariamente, referenciar uma entidade do tipo A.

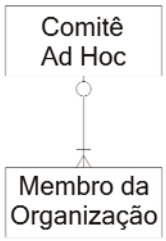



Fig. 5:
Membros de uma organização podem (mas não necessariamente) estar ativos em um comitê de trabalho. Um Comitê tem que ter (um ou mais) membros participando.

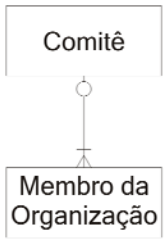



Fig. 6:
Membros de uma organização podem (mas não necessariamente) estar ativos em um comitê de trabalho. Um Comitê tem que ter (um ou mais) membros participando.

As figuras 5 e 6, tem modelo de dados similares, mas regras de negócio diferentes resultando em diferentes arquivos lógicos identificados.

**(In)Dependência de Entidade em um Relacionamento 1:N**

**1 : N**

Em um relacionamento 1:N entre duas entidades A e B, cada entidade B tem que ser atribuída a um e apenas um A, e a cada A tem que ser atribuído pelo menos a um B. Aplicam-se as mesmas regras de dependência e independência das entidades.


**Situação 1**

Se B *não* é significativo para o negócio independentemente do A a ele relacionado, então B é considerado uma entidade dependente de A.

**Situação 2**

Se B *é* significativo para o negócio independentemente do A a ele relacionado, então B é considerado uma entidade independente de A.

A APF conta as entidades A e B como um arquivo lógico na situação (1) (*dependência*), enquanto que na situação (2) A e B são arquivos lógicos separados (*independência*), como indicado na tabela da Seção “Resumo: de Entidades para Arquivos Lógicos via (In-)Dependência de Entidades”.

## Resumo: De Entidades para Arquivos Lógicos via (In)Dependência de Entidades

Na tabela abaixo, A e B são duas entidades de um modelo de dados (normalizado) que devem ser contadas de acordo com esta Seção e que são interconectadas via um relacionamento. A tabela resume como as diversas situações são contadas.

| Tipo de Relacionamento entre duas entidades, A e B | Quando esta Condição Existe | Então conte como Arquivo Lógico (AL) |
|---|---|---|
| (1) : (N) | (A e B são independentes) | 2 ALs |
| 1 : N | Se B é entidade dependente de A | 1 AL |
| | Se B é entidade independente de A | 2 ALs |
| 1 : (N) | Se B é entidade dependente de A | 1 AL |
| | Se B é entidade independente de A | 2 ALs |
| (1) : N | Se A é entidade dependente de B | 1 AL |
| | Se A é entidade independente de B | 2 ALs |
| (1) : (1) | (A e B são independentes) | 2 ALs |
| 1 : 1 | (A e B são dependentes) | 1 AL |
| 1 : (1) | Se B é entidade dependente de A | 1 AL |
| | Se B é entidade independente de A | 2 ALs |
| (N) : (M) | (A e B são independentes) | 2 ALs |
| N : M | Se B é entidade dependente de A | 1 AL |
| | Se B é entidade independente de A | 2 ALs |
| N : (M) | Se B é entidade dependente de A | 1 AL |
| | Se B é entidade independente de A | 2 ALs |

**Legenda**

AL = Arquivo lógico (ALI ou AIE)
(..) = Lado opcional do relacionamento

**Notas**

1. Na dúvida, decida por entidades independentes.

2. Em algumas situações mais que duas entidades podem também formar um arquivo lógico.

## Passo 2: Classificar Arquivos Lógicos

Os arquivos lógicos identificados precisam ser validados segundo as regras de contagem de ALI/AIE na Parte 1.

Classifique um arquivo lógico como um Arquivo Lógico Interno (ALI) se processos elementares dentro da fronteira da aplicação que está sendo contada, mantém (criam, alteram ou excluem) elementos de dados dentro do arquivo.

Classifique um arquivo lógico como um Arquivo de Interface Externa (AIE) se processos elementares dentro da fronteira da aplicação sendo contada apenas referenciam os elementos de dados dentro do arquivo, *e* o arquivo lógico é mantido por um processo elementar em outra aplicação.

Se um arquivo lógico identificado *não* é mantido por um processo elementar (dentro desta aplicação ou em outra), então o arquivo lógico não é contado de modo algum.

## Passo 3: Identifique Tipos de Dados Elementares

O dado elementar é a menor unidade que tem significado para o usuário e representa um fato específico sobre um negócio, por exemplo:

| Nome do Dado Elementar | Valor do Dado Elementar |
|---|---|
| **Taxa** | **$900** |
| **Data do Nascimento** | **15 Jan 1965** |
| **Nome** | **InfoMerge** |

## Termos e Definições de Dados Elementares

Ao iniciar o estudo de um modelo de dados lógico, começamos considerando esses elementos de dados como atributos. Um atributo representa um fato específico sobre uma entidade ou um relacionamento. Na tabela abaixo as entidades são mostradas em MAIÚSCULO, os atributos em minúsculo:

| Representação | Exemplo |
|---|---|
| **ENTIDADE_atributo** | **CURSO_taxa** |
| **ENTIDADE.atributo** | **COMPANHIACLIENTE.nome** |

Atributos/elementos de dados podem ser encontrados em:

- Visões do usuário (relatórios, telas)
- Dicionários de dados (modelos do negócio, modelos de dados)
- Arquivos existentes (estrutura de registros em programas, layouts de arquivos)

Quando estiver revendo os dados, o analista de dados segue esta premissa básica: todos os elementos de dados reconhecidos pelo usuário devem ser tratados como um atributo, e dessa forma devem ser mostrados em relação a uma entidade específica.

O atributo pode ter as seguintes propriedades: nome (sinônimo), característica, propósito (uso), origem, valores válidos, valor (estrutura), unidade, e dependências. Iremos explorar estas propriedades antes de rever o mapeamento de DERs para atributos/elementos de dados da Análise de Pontos de Função do IFPUG.

**Nome do Atributo**

Um nome único que resume as características apresentadas. Ele contém os seguintes componentes:

**origem** (entidade/relacionamento) *seguido por um ponto*

**descritivo** (adjetivo do atributo) *seguido por um hífen*

**classe** (atributo base)

Muitas tecnologias não aceitam espaços. Então vários nomes são concatenados com hífens.

Exemplo:

COMPANHIA_CLIENTE.endereço-entrega

SEMINARIO_MATRICULA. efetividade-avaliação

**Característica**

A propriedade do ambiente sendo medido ou representado. O nome da entidade que tem a característica é sempre presente; por exemplo, Endereço-Entrega: endereço em que os materiais serão entregues para a COMPANHIA-CLIENTE.

**Propósito**

Fazer a pergunta “Como o atributo é utilizado pelo negócio” justifica o atributo**.**

Exemplo: CURSO.data-qualificação

Propósito: Usado nas seções de recapitulação de planejamento

**Dependências**

As situações onde outros valores de atributos no modelo influenciam ou restringem o valor deste atributo. Por exemplo, CURSO-grau final não pode existir antes do término do curso, mas tem que existir ao término do mesmo.

**Atributos Chave**

Fornecem o relacionamento entre uma entidade e outra. Existem diferentes tipos de chaves no modelo de dados, ex.: chaves primárias, chaves secundárias e chaves estrangeiras.

Uma **Chave Primária (PK)** é o identificador único de uma entidade.

**Chaves Secundárias (SK)** são atributos que fornecem acesso mais rápido às informações, como:

LIVROESCOLAR.preço (SK)

LIVROESCOLAR.nome-editora (SK).

Chaves Secundárias não fazem parte da informação do modelo de dados (modelo de dados lógico) mas são usados principalmente para auxiliar no acesso (implementação física).

**Chaves Estrangeiras (FK)** são atributos usados para representar relacionamentos de uma entidade com outra.

**Atribuição**

O último conceito de modelagem de dados que devemos considerar antes da análise dos DERs é Atribuição, que prescreve/descreve onde os atributos residem, dentro da entidade ou dentro do relacionamento. Existem algumas regras comuns de atribuição que são seguidas na modelagem de dados:

1. Um atributo é atribuído à sua melhor "origem" única, que é indicada na propriedade de característica no FORMULÁRIO DE DEFINIÇÃO DE ATRIBUTO.
2. O identificador único (chave primária) de uma entidade será atribuído também a cada relacionamento em que ele participa.

Existem algumas diretrizes adicionais a serem seguidas ao tentar colocar um atributo na entidade mais apropriada:

1. Se a definição do atributo referir-se a uma entidade, aloque o atributo àquela entidade
2. Se a definição do atributo referir-se a diversas entidades, então crie um relacionamento e aloque o atributo ao relacionamento ou à entidade à qual ele se aplicar.

## Mapeando Elementos de Dados para Tipos de Elementos de Dados da APF

Agora que nós já revisamos os conceitos de elementos de dados e atributos na perspectiva da modelagem de dados, podemos relacionar estes conceitos às definições e regras de Pontos de Função do IFPUG:

| Conceito de Modelagem de Dados | Termo da Modelagem de Dados | Termo de BD Relacional | Termo da APF | Conceito da APF |
|---|---|---|---|---|
| Menor unidade de dados definida que tem significado no mundo real | Item de Dados | Atributo ou Coluna | Tipo de Elemento de Dados (DER - Dado Elementar Referenciado) | Um tipo de elemento de dados (DER) é um campo reconhecido pelo usuário, único e não repetido |
| Grupos de itens relacionados que são tratados como uma unidade | Registro | Linha ou Tupla | Tipo de Registro Elementar (RLR - Registro Lógico Referenciado) | Um tipo de registro elementar (RLR) é um subgrupo de elementos de dados, reconhecido pelo usuário, dentro de um ALI ou AIE |
| Coleção de registros de um mesmo tipo | Arquivo | Tabela | Arquivo Lógico (Arquivo Lógico Interno – ALI ou Arquivo de Interface Externa – AIE) | Arquivo se refere a um grupo de dados relacionados logicamente e não à implementação física deste grupo de dados |

**Tipos de Elementos de Dados (DERs)** são campos ou atributos, reconhecidos pelo usuário, únicos e não repetidos.

As seguintes regras se aplicam ao contar DERs em um arquivo lógico:

- Conte um DER para cada campo único, reconhecido pelo usuário e não repetido, mantido em/ou recuperado de uma função de dados através da execução de todos os processos elementares dentro de um

escopo de contagem

- Conte apenas aqueles DERs que estão sendo usados pela aplicação que está sendo contada quando duas ou mais aplicações mantém e/ou referenciam a mesma função de dados
- Conte um DER para cada atributo requerido pelo usuário para estabelecer um relacionamento com outra função de dados
- Revise atributos relacionados para determinar se são agrupados e contados como um único DER ou se são contados como vários DERs; o agrupamento vai depender de como o processo elementar utiliza os atributos dentro da aplicação

Não conte atributos que existam puramente para satisfazer um requisito técnico e não foram especificados pelo usuário. Exemplos destes atributos não funcionais são atributos resultantes de considerações de projeto ou de implementação.

Exemplo: O campo PEDIDO_data é contado como um DER no Pedido já que ele precisa ser mantido para satisfazer um requisito do negócio do usuário. Entretanto, a marca (stamp) de data e hora de cada registro do pedido existe para satisfazer a integridade e a confiabilidade dos dados. A solução técnica para estes requisitos de qualidade foi copiar o banco de dados e disponibilizar para recuperação baseado nesta informação da marca (stamp). Consequentemente, a marca (stamp) de data e hora não deve ser contada como um DER.

## Outras Situações

A seguir exemplos de contagem de tipo de elementos de dados (DERs).

## Atributos

Atributos que são compostos de diversos elementos de dados relacionados são armazenados separadamente.

Devem os atributos serem contados como diversos elementos de dados ou como um único elemento de dados? As regras de DER na Parte 1 dizem para você “contar cada campo reconhecido pelo usuário”.

Como você determina se ele é reconhecido pelo usuário como uma coisa ou várias coisas? Reveja as transações dentro da aplicação para determinar se o atributo é tratado como um item ou mais de um.

Considere os seguintes itens na tomada de decisão:

a) Se o atributo é sempre usado por inteiro, então ele é contado como um único elemento de dados (DER). Não devem existir situações em que um componente individual de um atributo é usado sem os outros. Baseado neste uso, o atributo é contado como um único elemento de dado.

b) Se em algumas situações, apenas uma parte do atributo (ex. o sobrenome) é usada, então mais do que um elemento de dados deve ser contado. Olhe para o uso em componentes dentro da aplicação para determinar quantas partes reconhecidas existem. A opção não é necessariamente um ou todos. Baseado no que você está vendo, pode ser apropriado contar apenas dois DERs, ainda que existam na realidade cinco partes físicas.

c) Olhe para a existência de requisitos de ordenação ou de edições e critérios de seleção. Se uma lista ou relatório é ordenado ou selecionado por um simples componente do atributo, isto sugere independência de componentes na visão do usuário.

## Contando Nomes

Nome (primeiro nome, nome do meio, último nome)

Muitas aplicações precisam manter informações sobre os nomes das pessoas.

O nome deve ser contado como vários elementos de dados ou um elemento de dados único?

Reveja as transações dentro da aplicação para determinar se o nome é tratado como um item ou mais do que um item. Por exemplo: Nos Estudos de Caso 1,2 e 3, veja como o Nome do Funcionário é usado em várias funções de transação.

Essas funções sempre usam o nome inteiro ou algumas vezes apenas usam uma parte?

Nos Estudos de Caso, o Nome do Funcionário é sempre usado inteiramente. Não existem telas ou relatórios onde uma única parte do nome é usada sem as outras partes. Também não existem situações onde uma parte do nome é usada para ordenação, edição ou critério de seleção. Seu uso dentro da aplicação sugere que Nome do Funcionário é um elemento de dados único (DER).

## Contando Endereços

Endereço (endereço, cidade, estado e CEP)

Muitas aplicações precisam manter informações sobre endereços. O endereço ser deve contado como vários elementos de dados ou um elemento de dados único?

Reveja as transações dentro da aplicação para determinar se o endereço é tratado como um item ou mais do que um item. Por exemplo: Nos Estudos de Caso 1, 2 e 3, veja como o Endereço de Localização é usado em várias funções de transação.

As transações sempre referenciam o endereço inteiro ou algumas vezes apenas usam uma parte?

Nos Estudos de Caso, o Endereço é sempre usado inteiramente. Não existem telas ou relatórios onde uma única parte do endereço seja usada sem as outras partes. Não existe situação onde uma parte do endereço seja usada para ordenação, edição ou critério de seleção. Seu uso dentro da aplicação sugere que Endereço da Localização é um elemento de dado único (DER).

Se existisse uma lista de locais que permitisse ao usuário listar todos os locais em uma cidade, estado ou CEP específico, mais de um elemento de dado deveria ser contado. Com base nas informações fornecidas, aparentemente cidade, estado e CEP são reconhecidos pelo usuário como partes independentes do endereço. Dessa forma, quatro DERs devem ser contados (Endereço, Cidade, Estado e CEP).

## Contando Campos Repetitivos

Muitas vezes as aplicações mantém múltiplas ocorrências de um elemento de dados. De acordo com as regras de DER na Parte 1, conte um campo repetitivo apenas uma vez.

Depois que os campos repetitivos forem reduzidos para um único DER, verifique se os requisitos do negócio estão ainda sendo satisfeitos. Considere os seguintes exemplos:

**Exemplo 1: Código do Funcionário**

Nos Estudos de Casos 1, 2 e 3, olhe para os requisitos de Manter Funcionário e para o relacionamento entre Funcionário e Dependente no Diagrama de Entidade e Relacionamento (DER). De acordo com os resultados dos Estudos de Casos, Funcionário é um AL que inclui Dependente. Os arquivos lógicos de Funcionário e Dependente conteriam cada um o Código do Funcionário.

Aplicando as regras de DER "repetitivos", conte Código do Funcionário como um único DER para o ALI Funcionário. Agora determine se os requisitos do negócio ainda são satisfeitos.

O requisito era manter Dependente como uma parte da informação do Funcionário. Sim, os requisitos de negócio estão satisfeitos, portanto, um DER é contado.

**Exemplo 2: Horas Trabalhada Diariamente**

Um sistema de reportar horas tipicamente mantém o número de horas que uma pessoa trabalha a cada dia. Ao rever as estruturas de dados, as Horas Trabalhadas são salvas separadamente para cada dia da semana (Horas Trabalhada de Segunda-feira, Horas Trabalhada de Terça-feira, etc.).

Aplicando as regras de DER "repetitivos", conte apenas Horas Trabalhadas. Determine se os requisitos do negócio ainda são satisfeitos. Se a aplicação apenas mantém informação sobre Horas Trabalhadas, está satisfeito o requisito do negócio para manter horas trabalhadas a cada dia?

Não, não está. A fim de satisfazer aquele requisito, conte Dia da Semana. A aplicação tem a habilidade de acompanhar separadamente cada hora trabalhada a cada dia (Horas Trabalhada de Segunda-feira, Horas Trabalhada de Terça-feira, etc.), portanto, dois DERs são contados.

## Contando Campos de Status

As aplicações frequentemente mantém informação do status atual dos dados (ex., Ativo, Inativo, Pendente, Aprovado, etc.). Este status é normalmente atualizado através das diversas transações dentro da aplicação. Estes campos de status podem ou não ser fisicamente visíveis ao usuário através das transações da aplicação. Considere os seguintes exemplos:

**Exemplo 1: Status Inativo**

Os Estudos de Caso 1, 2 e 3, incluem um indicador de Status. Quando um cargo ou um funcionário é excluído, os requisitos do usuário indicam que toda atribuição ao cargo excluído deve ser atualizada para o status de "inativo".

Ao rever as telas de atribuição de cargo em toda a aplicação, o status nunca é mostrado ou atualizado diretamente em nenhuma tela. Apesar da falta de visibilidade, o Status ainda é contado como um DER para o cargo atribuído. O fato dele aparecer no Modelo de Dados Lógico e nos Requisitos do Usuário sugerem que ele é reconhecido pelo usuário no Arquivo Lógico, mas em nenhuma transação. Então, um DER é contado no ALI.

**Exemplo 2: Status Não Contado**

O usuário solicita a habilidade de excluir Funcionários. A equipe técnica não quer fazer a exclusão física dos registros; então, eles implementaram um "flag de status" em Funcionário. Quando o usuário exclui um Funcionário, o Status é marcado como "Inativo". O usuário desconhece a existência do campo de Status em Funcionário. Então, o campo Status não deve ser contado como um DER.

## Contando Datas do Sistema

As aplicações frequentemente retêm datas do sistema associadas com seus dados para refletir a versão dos dados. Datas do Sistema podem ter muitos nomes diferentes (última atualização, última aprovação, etc.) e são frequentemente acompanhadas pelo código do usuário (última atualização por, última aprovação por). Estas datas do sistema são tipicamente atualizadas através de diversas transações dentro da aplicação. Em muitos casos, a data do sistema é mantida por requisitos de negócio. O usuário necessita saber quando o dado foi alterado ou aprovado. Existem também casos onde a data do sistema está sendo mantida apenas por razões técnicas.

Considere os seguintes exemplos:

**Exemplo 1: Data da Efetivação**

Os Estudos de Casos 1, 2 e 3, incluem uma referência à Data da Efetivação. Quando um cargo ou um funcionário é excluído, os requisitos do usuário indicam que qualquer atribuição de cargo associada deve ser atualizada para marcar a data da efetivação como sendo a data atual do sistema.

Ao rever as telas de atribuição de cargo em toda a aplicação, a data da efetivação nunca é mostrada ou atualizada diretamente em nenhuma tela. Apesar da falta de visibilidade, a Data da Efetivação é ainda contada como um DER para cargo. O fato dele aparecer no Modelo de Dados Lógico e os Requisitos do Usuário referenciarem-no pelo nome sugerem que o mesmo é reconhecido pelo usuário. Então, um DER é contado no Arquivo Lógico.

**Exemplo 2: Data da Recuperação**

A ferramenta de Backup/Recuperação utilizada pela aplicação usa a data do sistema armazenada na tabela para recuperar o dado para um ponto particular no tempo. Neste caso, a *Data da Recuperação* não é reconhecida pelo usuário e não deve ser contada.

**Exemplo 3: Data da Auditagem**

A equipe técnica decide que deve registrar a data do sistema e o código do usuário sempre que o dado é alterado para resolver qualquer questão futura sobre quando ou por quem uma alteração foi feita. Neste caso, a *data do sistema* não é reconhecida pelo usuário e não deve ser contada.

### Contando Chaves Estrangeiras

As aplicações frequentemente mantêm relacionamentos entre uma entidade e outra. Em alguns casos eles existem para satisfazer requisitos de validação de dados, mas em outros casos eles definem regras de negócio entre as duas entidades. O conceito de Atribuição na modelagem de dados é melhor ilustrado pela criação de chaves estrangeiras.

**Exemplo 1: Local (N) : (1)**

Os Estudos de Casos 1, 2 e 3 incluem os requisitos do usuário para Inclusão e Atualização de Funcionários: "O *local* deve ser um local válido no Sistema de Ativo Fixo (SAF)."

O Diagrama de Entidades e Relacionamentos também reforça este requisito do negócio pela ilustração do relacionamento entre a entidade Funcionário e a entidade Local. De acordo com o diagrama, um Funcionário pode ter um Local, e um Local pode ter muitos Funcionários. Nesta situação, o Nome do Local deve ser incluído nos atributos das tabelas lógicas e físicas. Conte o atributo Nome do Local como um elemento de dados (DER) para Funcionário.

**Exemplo 2: Cubículo**

Considere uma variação do exemplo anterior. Um empregado pode ter um número ilimitado de cubículos. Um cubículo só pode ser ocupado por um

**(1) : (N)**

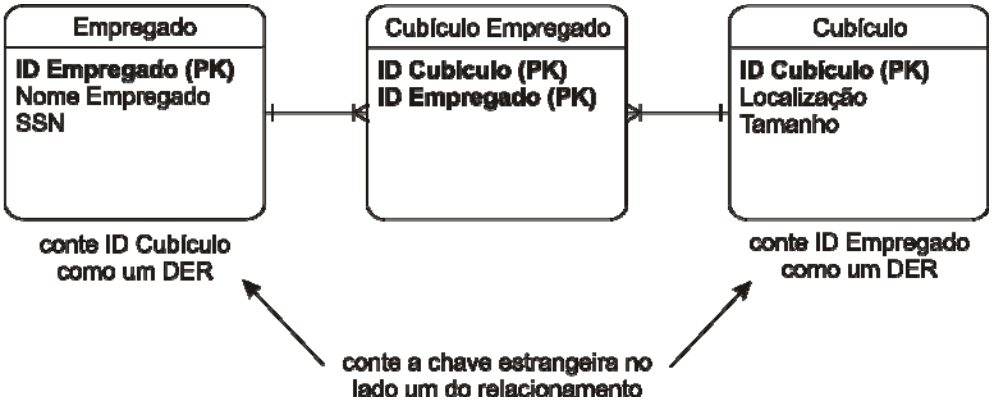
empregado por vez. O Código do Cubículo deve ser válido como identificado na tabela CUBÍCULO. O Diagrama de Entidades e Relacionamentos estaria garantindo o requisito do negócio pela ilustração do relacionamento entre a entidade CUBÍCULO e a entidade FUNCIONARIO. De acordo com o diagrama, um Empregado pode ter muitos cubículos, mas um Cubículo só pode ser ocupado por um Empregado. Nesta situação, o Código do Empregado é um atributo em CUBÍCULO.


Nota: nem todos os atributos estão retratados nas entidades.

O relacionamento deveria ser refletido na tabela CUBÍCULO pela identificação do Empregado que ocupa o Cubículo. O código do Empregado é contado como um elemento de dado (DER) para Cubículo.

**Exemplo 3 Cubículo (N) : (M)**

Considere uma variação do exemplo anterior. Um empregado tem que ter pelo menos um cubículo, mas pode ter um número ilimitado de cubículos. Um cubículo pode ser ocupado por mais de um empregado por vez. O Código do cubículo deve ser válido como identificado no CUBÍCULO. O Diagrama de Entidades e Relacionamentos novamente garantiria o requisito de negócio pela ilustração do relacionamento entre Empregado e Cubículo. De acordo com este diagrama, um Empregado pode ter muitos cubículos, e um Cubículo pode ser ocupado por muitos Empregados.




Nota: nem todos os atributos estão retratados nas entidades.

O relacionamento é demonstrado na tabela Cubículo-Empregado, o qual conteria uma ocorrência de cada empregado no relacionamento com o

cubículo. Ele incluiria o Código do Empregado e o Código do Cubículo como chave primária. Como explicado nas Seções “Passo 1: Identificar Arquivos Lógicos” e “Passo 4: Identificar Tipos de Registros Elementares”, Cubículo-Empregado não é contado como um Arquivo Lógico nem como um RLR. Código do Cubículo é contado como um elemento de dados (DER) em Empregado porque estabelece um relacionamento com a entidade Empregado e Código Empregado é contado como um elemento de dados (DER) em Cubículo porque estabelece um relacionamento com a entidade Cubículo.

### Observação

A identificação do número correto de DERs não influencia o número de arquivos lógicos, mas apenas sua complexidade. Enquanto, este passo influencia o tamanho funcional de forma limitada, o efeito é consideravelmente menor do que no Passo 1: Identificar Arquivos Lógicos.

## Passo 4: Identifique Tipos de Registro Elementares

O tipo de registro elementar (RLR) representa a visão do usuário dos *subgrupos* de dados dentro de um arquivo lógico identificado, o qual foi discutido enteriormente na Seção “Passo 1: Identificar Arquivos Lógicos”.

Tipos de registros elementares correspondem tipicalmente a entidades que foram agrupadas em arquivos lógicos como discutido na Seção “Passo 1: Identificar Arquivos Lógicos”. Eles precisam ser revistos cuidadosamente para garantir que o usuário os identifique como um subgrupo lógico, e deste modo, sejam contados como um Tipo de Registro Elementar (RLR).

### Termos e Definições de Tipo de Registro Elementar

Este capítulo mostra um modelo de dados lógico na 3ª. Forma Normal e ignora diversas entidades criadas por razões técnicas; se você não tem um modelo de dados uma tentativa deve ser feita para (des)normalizar os dados.

As definições estão baseadas nos conceitos de modelo de dados descritos na Seção “Conceitos de Modelagem de Dados”.

**Tipo de Entidade Associativa** Um tipo de entidade que contém atributos que descrevem em detalhe um relacionamento muitos-para-muitos entre dois outros tipos de entidades, também conhecida como entidade de intersecção.

**Tipo de Entidade Atributiva**

Um tipo de entidade que descreve em mais detalhes uma ou mais características de outro tipo de entidade.

**Entidade Subtipo**

Uma subdivisão de um tipo de entidade; herda todos os atributos e relacionamentos do seu tipo de entidade pai, e pode ter atributos e relacionamentos adicionais próprios.

**Mapeando os Termos de Modelagem de Dados para a Terminologia de Pontos de Função**

Os termos de Modelagem de Dados podem ser mapeados para a análise de pontos de função como mostrado na seguinte tabela:

| Conceito de Modelagem de Dados | Termo de Modelagem de Dados | Termo de Banco Dados Relacional | Termo da APF | Conceito de APF |
|---|---|---|---|---|
| Grupos de itens relacionados que são tratados como uma unidade | Registro | Linha ou Tupla | Tipo de Registro Elementar (RLR – Registro Lógico Referenciado) | Um *tipo de registro elementar* (RLR) é um subgrupo de elementos de dados, reconhecido pelo usuário dentro de um ALI ou AIE. |
| Coleção de registros de um mesmo tipo | Arquivo | Tabela | Arquivo Lógico (Arquivo lógico interno – ALI ou Arquivo de interface externa – AIE) | Arquivo se refere a um grupo de dados relacionados logicamente e não à implementação física deste grupo de dados. |

A seguinte tabela pode ser aplicada para auxiliar o entendimento do conceito de Tipo de Registro Elementar.

| Conceito de Entidades e Relacionamentos | Termo de Entidades e Relacionamentos | Termo da APF | Conceito de APF |
|---|---|---|---|
| Principais objetos de dados sobre os quais | Entidade ou Tipo de | Arquivo Lógico | Arquivo refere-se a um grupo de dados logicamente |

| | | | |
|---|---|---|---|
| informações são coletadas (pessoa, lugar, coisa ou evento); um item de fundamental importância para o usuário sobre os quais uma coleção de fatos é mantida. | Entidade | | relacionados e não à implementação física deste grupos de dados; se não existirem outros subgrupos, o arquivo lógico é contado com um único Tipo de Registro Elementar (RLR) |
| Um tipo de entidade que contém atributos que ajudam a descrever um relacionamento entre outras entidades. | Tipo de Entidade Associativa | Pode ser um arquivo lógico ou um possível tipo de registro elementar (RLR); veja a Seção "Analisando Entidades Associativas para determinar RLRs" para mais considerações. | Subgrupo de elementos de dados reconhecidos pelo usuário dentro de um ALI ou AIE, pode ser opcional ou obrigatório. |
| Um tipo de entidade que ajuda a descrever uma ou mais características de outro tipo de entidade. | Tipo de Entidade Atributiva | Possivelmente um Tipo de Registro Elementar (RLR); veja a Seção "Analisando Entidades Atributivas para Determinar RLRs" para mais considerações. | Subgrupo de elementos de dados reconhecidos pelo usuário dentro de um ALI ou AIE, pode ser opcional ou obrigatório. |
| Uma divisão do tipo de entidade, que herda todos os atributos e relacionamentos de seu tipo de entidade pai, e pode ter atributos e relacionamentos adicionais únicos. | Entidade Subtipo | Possivelmente um Tipo de Registro Elementar (RLR); veja a Seção "Analisando Subtipos para Determinar RLRs" para mais considerações. | Subgrupo de elementos de dados reconhecidos pelo usuário dentro de um ALI ou AIE, pode ser opcional ou obrigatório. |

A Análise de Pontos de Função considera as associativas, atributivas e subtipos como subgrupos de dados. Isto será explorado na discussão de como contar RLRs.

## Analisando Entidades Associativas para Determinar RLRs

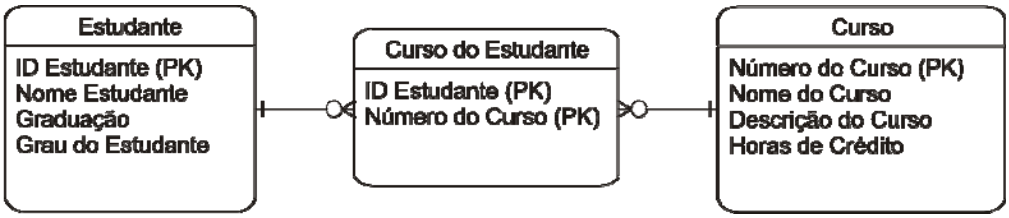
Uma entidade associativa é usada para associar duas ou mais entidades como uma maneira de definir um relacionamento muitos-para-muitos. Este tipo de entidade é frequentemente criada pelo modelador de dados para implementar algumas regras de negócio requeridas para relacionar duas entidades separadas.

Existem três possibilidades a considerar quando encontrar entidades associativas.

### Situação 1

Entidade Associativa *não é* contada como um RLR

A secretaria acadêmica tem um requisito de gerenciar todos os estudantes registrados para um curso e saber os cursos que o estudante completou previamente. Curso é uma entidade e Estudante é uma entidade. O modelador de dados criou uma entidade associativa chamada Curso do Estudante como uma interseção entre as duas, e esta entidade associativa apenas contém as chaves de cada entidade.



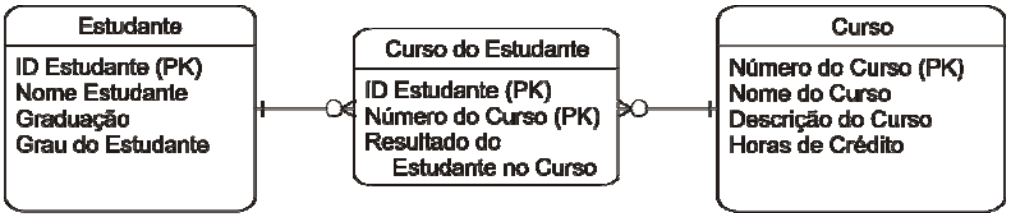

A entidade Curso do Estudante *não* é considerada um RLR nem deve ser contada como um arquivo lógico separado porque não contém nenhum elemento de dado adicional além das duas chaves primárias (PK) das entidades que fazem a interseção.

Estudante é um arquivo lógico com 1 RLR (Estudante) e Curso é um arquivo lógico com 1 RLR (Curso).

## Situação 2

Entidade Associativa *é* contada como um RLR

A secretaria acadêmica tem um requisito de identificar todos os estudantes registrados para um curso. Além disso ela precisa da informação sobre os resultados do curso para o(s) estudante(s). O Curso é uma entidade e o Estudante é uma entidade. O modelador de dados criou uma entidade associativa chamada Curso do Estudante como uma interseção entre as duas, e esta entidade associativa contém as chaves de cada entidade bem como o resultado do curso do estudante.




Não existe regra de negócio que solicite que curso do estudante seja mantido independentemente; então Curso do Estudante não satisfaz as regras para ser contado como um arquivo lógico separado. Neste caso, a entidade Curso do Estudante *é* considerada um RLR, pois contém pelo menos um atributo reconhecido pelo usuário (*), além das duas chaves primárias das entidades que participam da intersecção.

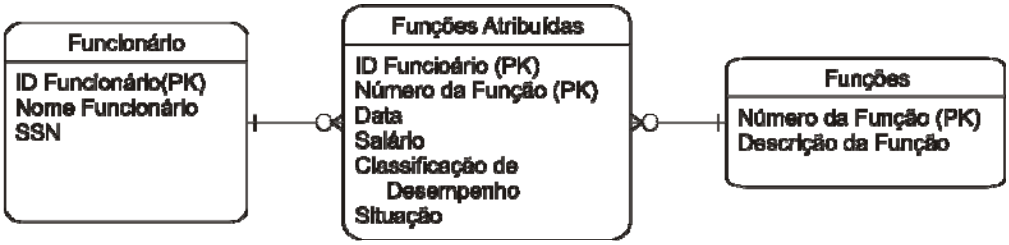
(*) Na Seção “Subpasso 1.5 Excluir entidades que não contém atributos requeridos pelo usuário”, atributos não chaves que são resultado de considerações de projeto ou de implementação, ou que satisfazem um requisito técnico, não são considerados elementos de dados.

A fim de ser contado como um RLR, um subgrupo deve conter um ou mais atributos além das chaves primárias. Estudante é um arquivo lógico com 2 RLRs (Estudante e Curso do Estudante) e Curso é um arquivo lógico com 2 RLRs (Curso e Curso do Estudante).

Se o requisito do negócio indicar que o relacionamento representado pela entidade associativa pertence a apenas um dos arquivos lógicos, o RLR será contado apenas naquele arquivo lógico.

**Situação 3** Entidade Associativa contada como um *arquivo lógico*, com um único RLR

Um departamento de RH mantém informações sobre Funcionário, Funções e Funções Atribuídas. A entidade Funções Atribuídas é necessária, mesmo que um Funcionário não esteja mais associado com a Função ou que a Função não seja mais uma função disponível para alocação.




Nota: Nem todos os atributos são retratados como entidades.

Embora Funções Atribuídas seja uma entidade associativa, ela é mais que um mapeamento key-to-key entre duas entidades, e é mais que um RLR associado com um arquivo lógico. Se uma regra de negócio solicita que a informação de Funções Atribuídas deva ser retida independentemente, Funções Atribuídas é considerada um arquivo lógico como descrito na Seção "Identificar Arquivos Lógicos Utilizando o Método de (In)Dependência de Entidades (Subpasso 1.3b)".

Se não existirem regras de negócio solicitando que as informações de Funções Atribuídas devam ser retidas independentemente, então esta entidade associativa é contada como na situação 2 acima.

## Analisando Entidades Atributivas para Determinar RLRs

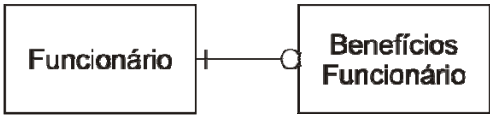
Uma entidade atributiva é um tipo de entidade que ajuda a descrever uma ou mais características de um outro tipo de entidade. Pela definição ela é uma extensão lógica de outra entidade; em Análise de Pontos de Função uma entidade atributiva representa um Tipo de Registro Elementar daquela entidade.

Uma entidade atributiva é contada como sendo um RLR do arquivo lógico que ela está definindo (Situação 1) ou como uma extensão do arquivo lógico (Situação 2).

### Situação 1

Uma entidade atributiva *opcional*

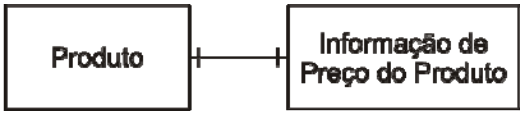
Um funcionário pode aderir a um plano de benefícios. Em nosso modelo de dados, Funcionário é uma entidade. Benefícios Funcionário é uma entidade atributiva de Funcionário e contém informações sobre os benefícios que o funcionário tem. Benefícios Funcionário não pode existir sem Funcionário, e é então logicamente relacionado.




Benefícios Funcionário é contado como um RLR pois é uma entidade atributiva opcional. Funcionário é um arquivo lógico com 2 RLRs, Funcionário e Benefícios de Funcionário.

**Situação 2** Uma entidade atributiva *obrigatória*

Um sistema de vendas deve manter informações sobre cada produto e sobre seus respectivos preços. Produto é uma entidade. Informações de Preço do Produto é uma entidade atributiva relacionada a Preço, contendo: preço anterior, preço atual, preço futuro projetado e data efetiva do preço. Informações de Preço do Produto não existe sem Produto e é então logicamente relacionada.




Informações de Preço do Produto *não* é contado como um RLR. Produto é um arquivo lógico com 1 RLR, contendo Produto e Informações sobre Preço do Produto.

## Analisando Subtipos para Determinar RLRs

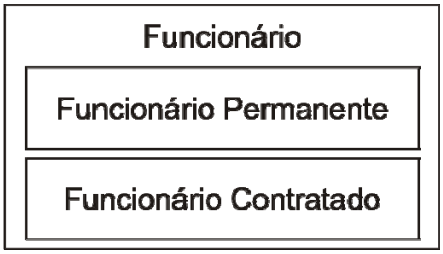
Uma entidade subtipo é uma subdivisão de tipo de entidade. Um subtipo herda todos os atributos e relacionamentos da entidade pai, e pode ter atributos e relacionamentos adicionais próprios. As regras de modelagem de dados determinam que uma entidade pode ter qualquer número de grupos de subtipos independentes associados a ela, os quais podem ser opcionais ou obrigatórios. Cada subtipo pode ter apenas um pai. Na modelagem de dados, embora o pai e o subtipo sejam representados como entidades diferentes, eles são logicamente parte da mesma entidade.

Ao analisar subtipos no modelo de dados, olhe para o relacionamento requeridos para a entidade "pai", como mostrado nas seguintes situações.

### Situação 1

Subtipo que é um subgrupo e então *é* contado como um RLR

Um funcionário tem que ser um funcionário permanente ou um funcionário contratado, mas não pode ser os dois. Os dados comuns do funcionário são pertinentes a todos os funcionários e são obrigatórios. Além disso, os dados comuns são herdados pelas entidades subtipo obrigatórias, permanente e contratado.




Na revisão destes dados a partir da perspectiva de pontos de função, dois subgrupos lógicos do arquivo lógico Funcionário são identificados:

- Dados do funcionário permanente incluem informações do funcionário permanente como também as informações comuns do funcionário.
- Dados do funcionário contratado incluem informações do funcionário contratado como também as informações comuns do funcionário.

Nesta situação existe um arquivo lógico (Funcionário) com dois RLRs, funcionário permanente e funcionário contratado.

**Situação 2** Subtipo que *não é* um subgrupo e então não é contado como um RLR

Se existem atributos únicos entre entidades subtipo, considere seriamente se um subgrupo separado realmente existe e desta forma constituiria um tipo de registro elementar (RLR). Um simples atributo opcional único não resultaria em um RLR diferente da perspectiva do usuário, mesmo se representado como uma entidade subtipo em um modelo de dados lógico.

O estado civil de um Funcionário pode ser casado ou solteiro. Se casado, o nome do cônjuge é armazenado. Embora possa ser representado como um subtipo em um modelo de dados, o nome do cônjuge é apenas um atributo opcional dentro do grupo lógico de dados do Funcionário.


Um atributo diferente neste caso não faz diferença significativa entre o funcionário casado e solteiro a partir da visão do negócio.

**Dicas** Olhe para o modelo de dados cuidadosamente. Quando houver dúvida, pergunte ao usuário a intenção dos subtipos separados. O analista de dados cria o modelo de dados representando sua visão do mundo do usuário. Na prática, depende da visão do usuário/regras de negócio se estas entidades subtipo são importantes para o usuário e devem ser consideradas como RLRs.

Se existem transações separadas para incluir/alterar atributos únicos para estas entidades subtipo, isto seria uma indicação de que *deveríamos* ter RLRs separados para estes subtipos de entidades.

## Outras Situações

Se você não tem um modelo de dados, procure grupos repetitivos de dados. Você pode encontrar algumas das seguintes situações. Aqui estão algumas dicas adicionais para contagem.

**Grupos /Dados Repetitivos**

Grupos repetitivos são múltiplas ocorrências dos mesmos dados, que podem ser repetidos diversas vezes dentro de um arquivo lógico.

**Situação 1**

*Grupos* repetitivos contados como RLR

O grupo de dados de Pedido consiste em Cabeçalho do Pedido e pode ter várias ocorrências de Item de Pedido. Item de Pedido contém mais do que um atributo único. Cabeçalho do Pedido e Item de Pedido representam dois subgrupos separados. Nós podemos contar dois RLRs para o arquivo lógico Pedido.

**Situação 2**

*Dados* repetitivos não contados como RLR

Um campo repetitivo (DER) não resultaria em um subgrupo separado ou RLR. Por exemplo, um Funcionário deve ter diversos números de contas de bancos. Isto *não* implicaria em dois RLRs para Funcionário ("todos os dados sem numero da conta do banco" e "números das contas dos bancos").

**Observação**

Na dúvida, *não* conte um subgrupo de informações como um RLR.

A Identificação do número correto de RLRs não influencia o *número* de arquivos lógicos identificados, influencia apenas na complexidade do arquivo lógico. Embora este passo influencie no tamanho funcional, esta influência é em nível inferior a dos arquivos lógicos no Passo 1: "Identificar Arquivos Lógicos".

## Considerando Tipo de Dados Elementares e Tipo de Registros Elementares em Conjunto com Arquivos Lógicos via (In)Dependência de Entidades

Agora que os Tipos de Dados Elementares e Tipos de Registros Elementares foram discutidos, a tabela mostrada na Seção "Resumo: de Entidades para Arquivos Lógicos via (In)Dependência de Entidades" é expandida, incluindo DERs e RLRs.

| Tipo de Relacionamento entre duas entidades, A e B | Quando esta Condição Existe | Então conte como Arquivos Lógicos com RLRs e DERs como abaixo: |
|---|---|---|
| (1) : (N) | (A e B são independentes) | 2 ALs, 1 RLR e DERs para cada |
| 1 : N | Se B é entidade dependente de A | 1 AL, 2 RLRs, soma de DERs |
| | Se B é entidade independente de A | 2 ALs, 1 RLR e DERs para cada |
| 1 : (N) | Se B é entidade dependente de A | 1 AL, 2 RLRs, soma de DERs |
| | Se B é entidade independente de A | 2 ALs, 1 RLR, e DERs para cada |
| (1) : N | Se A é entidade dependente de B | 1 AL, 2 RLRs, soma DERs |
| | Se A é entidade independente de B | 2 ALs, 1 RLR, e DERs para cada |
| (1) : (1) | (A e B são independentes) | 2 ALs, 1 RLR, e DERs para cada |
| 1 : 1 | (A e B são dependentes) | 1 AL, 1 RLR, soma DERs |
| 1 : (1) | Se B é entidade dependente de A | 1 AL, 1 ou 2 RLRs, soma DERs |
| | Se B é entidade independente de A | 2 ALs, 1 RLR, e DERs para cada |
| (N) : (M) | (A e B são independentes) | 2 ALs, 1 RLR, e DERs para cada |
| N : M | Se B é entidade dependente de A | 1 AL, 2 RLRs, soma DERs |
| | Se B é entidade independente de A | 2 ALs, 1 RLR, e DERs para cada |
| N : (M) | Se B é entidade dependente de A | 1 AL, 2 RLRs, soma DERs |
| | Se B é entidade independente de A | 2 ALs, 1 RLR, e DERs para cada |

## Notas

- 1 RLR e DERs para cada significa: avaliar as duas entidades por conta própria.
- Soma DERs significa: contar todos os atributos únicos, não repetidos de entidades ligadas entre si.
- Contar a chave estrangeira do lado muitos do relacionamento.
- Em algumas situações mais de duas entidades podem formar um arquivo lógico; nesse caso mais de dois (2) RLRs devem ser contados.

## Legenda

AL = Arquivo lógico (ALI ou AIE)
(..) = Lado opcional do relacionamento
RLR = Tipo de Registro Elementar
DER = Tipo de Dado Elementar

# Bibliografia

As fontes foram consultadas ou citadas neste capítulo.

Booch, Grady, James Rumbaugh, Ivar Jacobson. The Unified Modeling Language User Guide. Reading: Addison-Wesley, 1994. ISBN: 0-2015-7168-4.

NESMA. Definitions and Counting Guidelines for the Application of Function Point Analysis: A Practical Manual, Version 2.2. (NESMA, 2003).

ISBN: 978-90-76258-17-1.

Nota: Este manual é também chamado de NESMA Counting Practices Manual. Descreve o padrão da metodologia de APF, e muitos aspectos relacionados a aplicação de APF. Pode ser usado junto com o manual do IFPUG. Para maiores informações, acesse o site da NESMA www.nesma.org .

Garmus, David, David Herron. Function Point Analysis: Measurement Practices for Successful Software Projects. Boston: Addison-Wesley Information Technology Series, 2001. ISBN: 0-201-69944-3.

Martin, James, Carma McClure. Diagramming Techniques for Analyst and Programmers. Englewood Cliffs: Prentice-Hall, Inc., 1985. ISBN: 0-132-087944.

Modern Language Association of America. MLA Handbook for Writers of Research Papers, Fifth Edition. Boston: Addison Wesley, 1999.

Reingruber, Michael C. and William W. Gregory. The Data Modeling Handbook: A Best- Practice Approach to Building Quality Data Models. Canada: John Wiley & Sons, Wiley-QED Publication, 1994. ISBN: 0-471-05290-6.

Silverman, Len, W. H. Inmon, Kent Graziano. The Data Model Resource Book: A Library of Logical Data Models and Data Warehouse Design. Boston: Addison-Wesley, Inc. Out of Print: AISN: 0-471-15364-8.

Simsion, Graeme. Data Modeling Essentials: Analysis, Design, and Innovation. Boston: International Thomson Computer Press, 1994. ISBN: 1-850-932877-3.

Schuldt, Gary. "Information Modeling for Information Systems Analysts", A workshop at AT&T Bell Laboratories. Holmdel, N.J., May, 1992.

Teorey, Toby J. Database Modeling & Design: The Fundamental Principles, Second Edition. San Francisco: Morgan Kaufmann Publishers, Inc., 1994. ISBN: 1-558-60291-1.

Esta página foi deixada em branco intencionalmente.

# Parte 3 Capítulo 3

# Dados Compartilhados

**Introdução** Este capítulo fornece diretrizes adicionais para auxiliar na identificação de arquivos de interface externa (AIEs) e de arquivos lógicos internos (ALIs) bem como arquivos de transação quando dois ou mais sistemas interagem (isto é, diretrizes e esclarecimentos na contagem de dados que são compartilhados entre os sistemas).

**Conteúdo** Este capítulo inclui as seguintes seções:

# Contagem de Dados Compartilhados Entre Sistemas

### Aplicações que Compartilham Dados

As aplicações que compartilham dados com outras aplicações:

- Referenciam ou utilizam os dados para concluir uma transação que está sendo processada dentro do sistema que está recebendo ou acessando os dados, ou
- Mantêm arquivos lógicos internos dentro do sistema que está recebendo ou acessando os dados

### Métodos de Compartilhamento de Dados

Os dados compartilhados, utilizados pelos processos elementares dentro de uma aplicação para manter dados em um arquivo lógico interno ou para apresentar dados ao usuário, podem ser transferidos via:

- Telas on-line (ex. *screen scraping*)
- Acesso direto aos arquivos de dados de outros sistemas
- Arquivos transferidos
- Recuperação direta on-line real-time das informações
- Aplicações *web*

Com o intuito de analisar corretamente estas implementações, os usuários precisam considerar a intenção primária e ter um entendimento comum dos termos que representam as diversas implementações técnicas.

### Intenção Primária

O conceito de intenção primária é útil na identificação de arquivos lógicos internos e arquivos de interface externa com relação à utilização dos dados na aplicação sendo analisada. A intenção primária refere-se ao papel mais significativo ou importante que a função tem a intenção de realizar. A definição de intenção primária é "intenção que é o primeiro lugar em importância". Portanto, é importante **determinar a intenção primária** na discussão de cada cenário. A implementação física não afeta a intenção primária e deste modo não deve influenciar a análise.

### Definição & Intenção Primária (ALI/AIE)

**Arquivo Lógico Interno:**

Um Arquivo Lógico Interno (ALI) é um grupo de dados ou informações de controle logicamente relacionados, identificável pelo usuário, mantido dentro da fronteira da aplicação. A intenção primária de um ALI é armazenar dados mantidos através de um ou mais processos elementares da aplicação sendo contada.
Nota: O termo mantido é a capacidade de modificar dados através de um processo elementar. Exemplos incluem, mas não se limitam a, incluir, alterar, excluir, popular, revisar (corrigir), atualizar, assinalar e criar.

**Arquivo de Interface Externa:**

Um Arquivo de Interface Externa (AIE) é um grupo de dados logicamente relacionados ou informação de controle, reconhecido pelo usuário, referenciado pela aplicação sendo medida, mas que é mantido dentro da fronteira de outra aplicação. A intenção primária de um AIE é armazenar dados referenciados por um ou mais processos elementares dentro da fronteira da aplicação medida. Isto significa que um AIE contado por uma aplicação deve ser um ALI em outra aplicação.

## Termos Comuns

Os seguintes termos comuns são utilizados neste documento para descrever técnicas de implementações físicas:

| Termo | Utilizado no Documento |
|---|---|
| **Cópia** | Definição IEEE:<br>(1) Ler os dados de uma origem, deixando a fonte de dados inalterada, e gravar o mesmo dado em outro lugar em uma forma física que pode ser diferente daquela utlizada na fonte. Por exemplo, copiar dados de um disco magnético para uma fita magnética.<br>(2) O resultado de um processo de cópia como o descrito acima. Por exemplo, uma cópia de um arquivo de dados. |
| **Arquivo** | Definição IEEE:<br>"...grupo de registros relacionados tratados como uma unidade. Por exemplo, um arquivo pode consistir de um grupo de registros de fatura." |
| **Imagem** | Uma replicação exata de outro objeto, arquivo ou tabela normalmente criada através de um utilitário. |
| **Carga** | Definição IEEE:<br>"... para copiar instruções do computador ou dados de um depósito externo para um depósito interno ..." |
| **Merge** | Vários arquivos com os mesmos elementos de dados consolidados em um único arquivo. |
| **Refresh** | Processo de recriação de um grupo de dados para atualização a partir da origem. |

## Organização dos Cenários da Contagem

A APF muitas vezes se baseia nas descrições de desenvolvedores das características físicas de uma aplicação. Este capítulo trata estas descrições físicas que um analista de pontos de função frequentemente encontra para auxiliar na correta interpretação de muitas destas interfaces de aplicação.

Este capítulo utiliza diversos cenários como ajuda na análise de situações de dados compartilhados.

### Abordagem

Este capítulo usa as seguintes abordagens na discussão de dados compartilhados:

- ***Descrição*** - Uma expressão de alto nível do exemplo que está sendo discutido.
- ***Cenário*** - É apresentado um exemplo que geralmente descreve uma atividade física ou transação a respeito de arquivos sendo transferidos entre duas aplicações; ex., compartilhamento de dados.
- ***Diagrama do Cenário*** - O cenário é representado graficamente, como uma ajuda para mapear uma situação ou cenário similar. A seta nos diagramas reflete a direção do fluxo de dados, não a aplicação que inicia a interface.
- ***Interpretação da Contagem*** - Uma interpretação da contagem para o cenário é fornecida, que inclui uma discussão do exemplo e como o mesmo deve ser contado, bem como qualquer premissa com relação à intenção primária.
- ***Diagrama da Solução*** - A solução é representada graficamente.
- ***Resumo da Contagem*** - Os dados e funções de transação aplicáveis a cada aplicação são resumidos na tabela.
- ***FAQs/Variações*** - Se aplicável, algumas variações comuns do cenário podem ser incluídas na sequência da discussão.

### Símbolos Utilizados no Diagrama de Solução

Os seguintes símbolos são utilizados no diagrama de solução:

☑ acima de um tipo de componente indica que o componente **é contado** para a aplicação;

☒ acima de um tipo de componente indica que o componente **não deve ser contado** para o cenário

⇐ o diagrama retrata a direção do fluxo de dados, não a aplicação que inicia a interface

## Convenções de Nome dos Cenários

Para manter a consistência, as seguintes convenções para nomenclatura foram utilizadas em todos os cenários:

| Termo | Descrição |
|---|---|
| **Sistema A** | Sistema origem para os dados referenciados ou de transação. |
| **Sistema B** | Sistema que recebe os dados referenciados ou de transação. |
| **Arquivo X** | Um ALI contado no sistema A. |
| **Arquivo X (Principal X)** | Um AIE contado no Sistema B que é um subgrupo de dados do Arquivo X. |
| **Arquivo Y** | Um ALI contado no Sistema B. |
| **Arquivo Z** | Um arquivo de transferência de dados. Este arquivo é gerado pelo Sistema A e lido (processado) pelo Sistema B. |
| **Fronteiras** | As aplicações A e B representam duas aplicações separadas, assim, representam duas fronteiras separadas. |

## Resumo dos Cenários

Os cenários a seguir não representam uma lista completa das diversas formas que os dados compartilhados são implementados, mas fornecem diretrizes para muitas situações encontradas. A compreensão destes exemplos facilitará o entendimento de cenários adicionais que podem ser encontrados.

Os cenários focam situações onde os dados solicitados para completar os processos elementares da Aplicação B são obtidos da Aplicação A. A Aplicação B é a aplicação que está sendo contada. Os cenários são divididos em dois grupos, cada um dos quais possui diversas implementações:

**GRUPO 1:** A intenção primária é a Aplicação B referenciar dados mantidos pela Aplicação A. Existem duas áreas que são tratadas: funcional e não-funcional.

**Funcional** Por Razões Funcionais (requisitos do sistema), os sistemas compartilham dados nos seguintes cenários

| Número do Cenário | Cenário | Resumo da Descrição |
|---|---|---|
| **1** | **LEITURA** | A Aplicação B acessa fisicamente os dados da Aplicação A. Este exemplo está atualmente documentado no CPM. |

| Número do Cenário | Cenário | Resumo da Descrição |
|---|---|---|
| **2** | **CÓPIA ESTÁTICA DE IMAGEM** | A Aplicação A gera uma imagem de um depósito de dados, que reflete o estado atual dos dados em um certo tempo e permanece dentro desta fronteira. |

**Não-Funcional** Por Razões Não-Funcionais (performance, segurança, etc.), a Aplicação B deve usar os dados da Aplicação A e o faz da seguinte forma:

| Número do Cenário | Cenário | Resumo da Descrição |
|---|---|---|
| **3** | **CÓPIA/CARGA DE IMAGEM**<br>**Sem Lógica de Processamento** | A Aplicação A gera uma imagem sem lógica de processamento adicional e a envia para a Aplicação B; o Sistema B carrega a cópia sem lógica de processamento adicional. |
| **4** | **CÓPIA/CARGA DE IMAGEM**<br>**Subgrupo de um ALI** | A Aplicação A gera uma cópia exata de um subgrupo (ex. RLR) sem lógica de processamento adicional e a envia para a Aplicação B. A Aplicação B carrega o RLR sem lógica de processamento adicional. |
| **5** | **CÓPIA / MERGE**<br>**"*Refresh*"** | Os dados guardados em dois sistemas são copiados e mesclados para formar um arquivo que é carregado num terceiro sistema. |
| **6** | **SCREEN SCRAPING** | A Aplicação B acessa telas da Aplicação A para referenciar/obter dados para uso no processamento de uma transação. |

**GRUPO 2** A intenção primária é que a Aplicação B mantenha seus próprios dados através dos dados mantidos pela Aplicação A.

| Número do Cenário | Cenário | Descrição do Cenário |
|---|---|---|
| **7** | **MANTER DEPÓSITO DE DADOS COMUM** | O mesmo depósito de dados é mantido por duas aplicações diferentes. Este exemplo está atualmente documentado no CPM. |
| **8** | **DADOS PADRÃO DE TRANSAÇÃO** | Dados de transação são fornecidos pela aplicação de origem. |

# Cenários de Contagem – Grupo 1

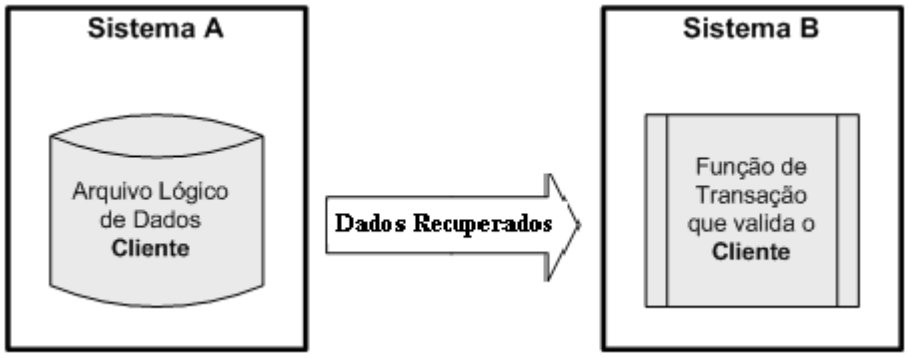
Em cada um dos cenários de 1 a 6, a intenção primária é uma aplicação referenciar dados mantidos por uma ou mais aplicações diferentes; isto pode ser implementado das seguintes maneiras:

## Cenário 1: Leitura

**Descrição** A Aplicação B acessa fisicamente os dados da Aplicação A para executar uma consulta.

**Cenário** Uma transação processada pela Aplicação B precisa de informações de um depósito de dados mantido dentro da Aplicação A. A Aplicação B é responsável pelo acesso aos dados da Aplicação A, e a Aplicação B mantém o software para este acesso.

**Diagrama**

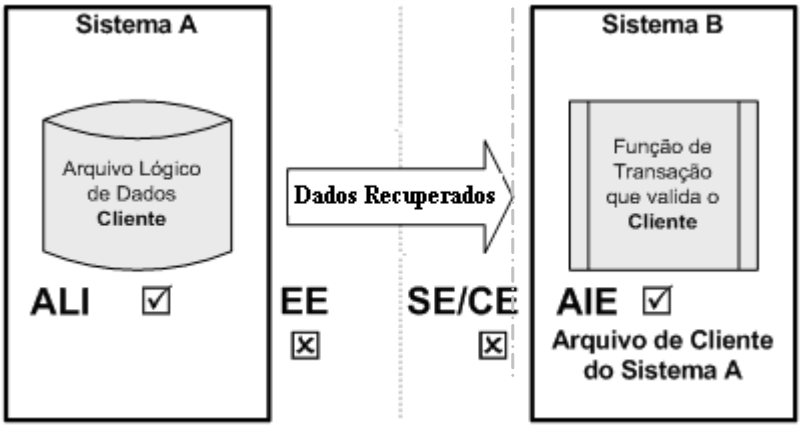



**Interpretação de Contagem**

***Aplicação A*:** Na perspectiva da Aplicação A, não existe requisito para enviar dados. Os dados estão disponíveis na Aplicação A. Nenhum crédito é dado para a Aplicação A para a transação executada pela Aplicação B, embora o arquivo de dados seja um ALI para a Aplicação A.

***Aplicação B***: Na perspectiva da Aplicação B, tanto logicamente quanto fisicamente, existe apenas um depósito de dados envolvido. A Aplicação B conta o depósito de dados que reside na Aplicação A como um AIE. A Aplicação B também conta aquele arquivo de dados como um ALR na transação.

**Diagrama da Solução**




**Resumo da Contagem**

| | ALI | AIE | EE | SE/CE | Nota |
|---|---|---|---|---|---|
| Aplicação A | ☑ | | | | |
| Aplicação B | | ☑ | | | Cliente é também contado como um ALR na função de transação. |

## Cenário 2: Cópia Estática de Imagem

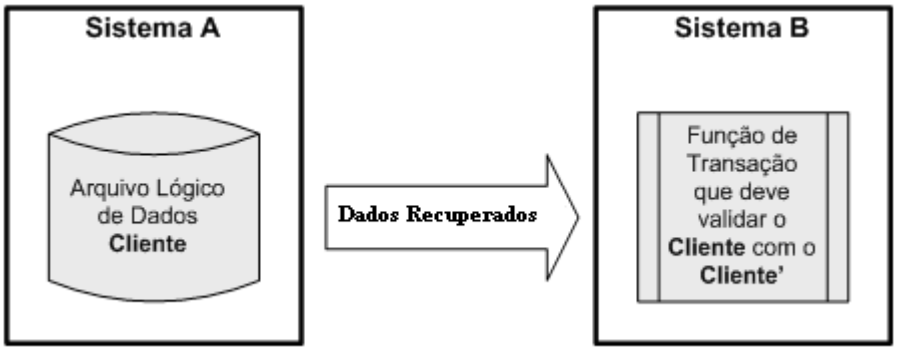
**Descrição**

A Aplicação A gera uma cópia estática de um ALI, refletindo o estado atual dos dados daquele momento, e a cópia permanece em sua fronteira.

**Cenário**

No setor bancário, as transações financeiras são conciliadas diariamente entre todas as instituições financeiras. As transações financeiras subsequentes dos clientes são validadas contra o respectivo saldo a partir desta conciliação. Para atender a este requisito do negócio, a Aplicação A periodicamente gera uma cópia estática dos dados do arquivo lógico Cliente para o Cliente principal (ou Cliente') para que outros sistemas possam referenciá-lo. Cliente' (ou Cliente principal) permanece dentro da fronteira da Aplicação A. Podem existir diferenças entre os dados atuais do Cliente e os dados do Cliente'. A Aplicação B utiliza Cliente'.

**Diagrama de Solução**

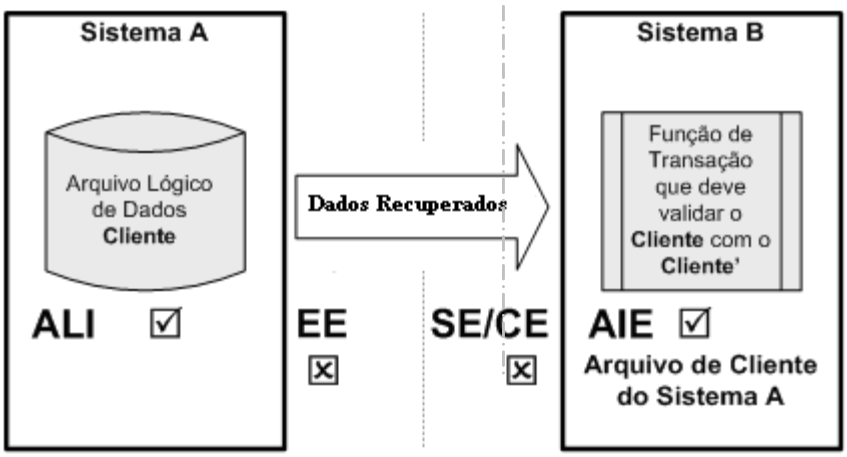



**Interpretação da Contagem**

***Aplicação A***: Na perspectiva da Aplicação A, Cliente é um arquivo lógico interno para a Aplicação A. O Cliente' (principal) não é contado como um ALI separado, nem é tampouco contado como um RLR do Cliente. Cliente' é apenas uma fotografia de Cliente em um determinado tempo.

***Aplicação B***: Na perspectiva da Aplicação B, Cliente é um arquivo de interface externa para a Aplicação B e é também contado como um ALR para a transação da Aplicação B.

**Diagrama da Solução**




**Resumo da Contagem**

| | ALI | AIE | EE | SE/CE | Nota |
|---|---|---|---|---|---|
| Aplicação A | ☑ | | | | |
| Aplicação B | | ☑ | | | Cliente é também contado como um ALR na função de transação. |

## Cenário 3: Cópia/Carga de Imagem - Nenhum Processamento Adicional

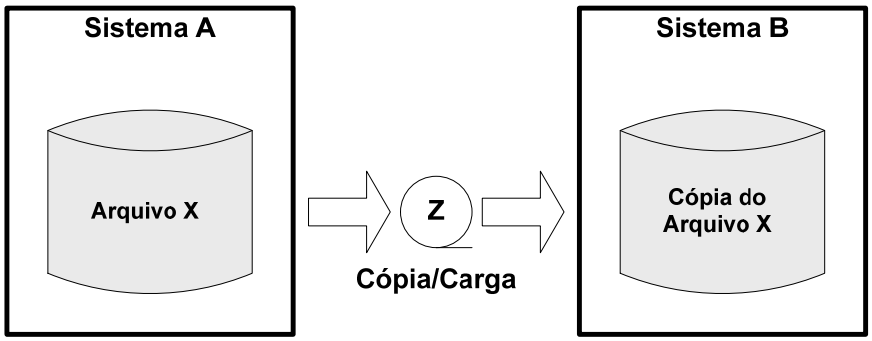
**Descrição**

A Aplicação A gera uma cópia estática sem lógica de processamento adicional e a envia à Aplicação B; a Aplicação B carrega uma cópia sem lógica de processamento adicional

**Cenário**

A Aplicação B requer a habilidade de acessar o arquivo X na Aplicação A apenas para validação e referência. A Aplicação B requer (ex. performance, etc) que a Aplicação A envie um arquivo completo para a Aplicação B. Os dados existentes armazenados na Aplicação B são atualizados a cada vez com a cópia.

**Diagrama**




**Interpretação de Contagem**

Na perspectiva da Aplicação A, esta transferência de dados é uma solução técnica criada para satisfazer o requisito de negócio em que a Aplicação B deve ter acesso, com o propósito de recuperação de dados, ao Arquivo X da Aplicação A.

Logicamente os dados armazenados permanecem na Aplicação A. Neste caso, a cópia dos dados armazenados de uma aplicação para outra é a solução de um requisito não-funcional do usuário (por ex., os dados na Aplicação A não estão disponíveis quando são solicitados pela Aplicação B).

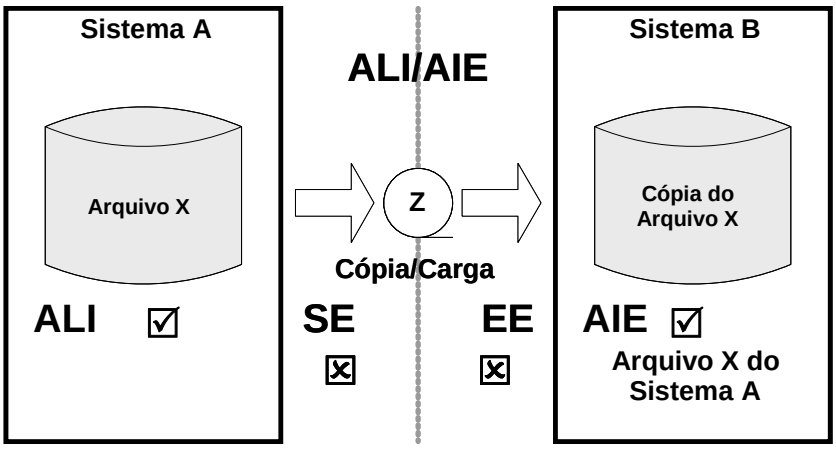
A *intenção primária* da perspectiva de B é referenciar os dados que estão logicamente em A. Uma indicação adicional é que o arquivo no Sistema B é "atualizado" a cada vez com a cópia. Da mesma forma, nenhuma lógica de processamento é executada nem na Aplicação A nem na Aplicação B.

*Transações* - As transações para transferência de dados: o download da aplicação A e a carga do arquivo pela Aplicação B são partes de uma solução técnica e não são contados em nenhuma das aplicações. Na prática, ao contar

a Aplicação A isoladamente, pode não ficar aparente para o Analista de Pontos de Função que esta solução exista para satisfazer um requisito não-funcional do usuário, e a mesma pode ser contada incorretamente como uma CE/SE. Nenhum dos dois sistemas conta o Arquivo Z como uma função de transação.

*Arquivos* - Existe apenas um arquivo lógico envolvido. A Aplicação A conta o Arquivo X como um ALI. A Aplicação B conta sua versão copiada do Arquivo X como um AIE. Nenhum dos dois sistemas conta o Arquivo Z como uma função de dados.

**Diagrama da Solução**




**Resumo da Contagem**

| | ALI | AIE | EE | SE/CE | Nota |
|---|---|---|---|---|---|
| Aplicação A | ☑ | | | | |
| Aplicação B | | ☑ | | | |

**FAQs, Variações Adicionais**

**P?** O que acontece se um arquivo lógico na Aplicação A é composto por diversas tabelas físicas e a Aplicação A fornece cópias individuais de mais de uma tabela para a Aplicação B?

**R:** O AIE é identificado da mesma maneira como no cenário acima; apenas os campos usados são contados como DERs.

**P?** O que acontece se você tem um armazenamento de dados particionado?

**R:** É particionado para melhor performance, mas ainda é logicamente um arquivo de dados; representa implementação física.

## Cenário 4: Cópia/Carga de Imagem de uma Tabela Física – Nenhum Processamento Adicional

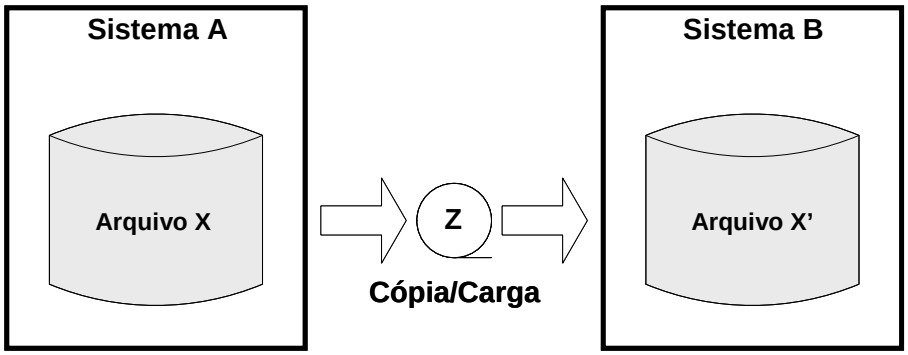
**Descrição**

A Aplicação A gera uma cópia de uma tabela física dentro de um arquivo lógico da Aplicação A sem lógica de processamento adicional e a envia à Aplicação B. A Aplicação B carrega a tabela física sem qualquer lógica de processamento adicional.

**Cenário**

A Aplicação B requer (p. ex., performance, etc.) a habilidade de acessar uma parte do arquivo X na Aplicação A apenas para validação e referência. A Aplicação A envia uma tabela física com o arquivo lógico para a Aplicação B. A visão existente daquela tabela física na Aplicação B é "atualizada" a cada vez com a cópia.

**Diagrama**




**Interpretação de Contagem**

Uma vez que o dado é uma cópia da imagem dos dados da Aplicação A, a tabela do Arquivo X' é parte do arquivo lógico X da Aplicação A. A Aplicação B conta o Arquivo X' (com apenas os elementos de dados usados da tabela do Arquivo X') como um AIE.

Logicamente os dados armazenados permanecem na Aplicação A. Neste caso, a cópia dos dados armazenados de uma aplicação para outra é a solução de requisitos não-funcionais do usuário; p. ex., os dados na Aplicação A não estão disponíveis quando solicitados pela Aplicação B.

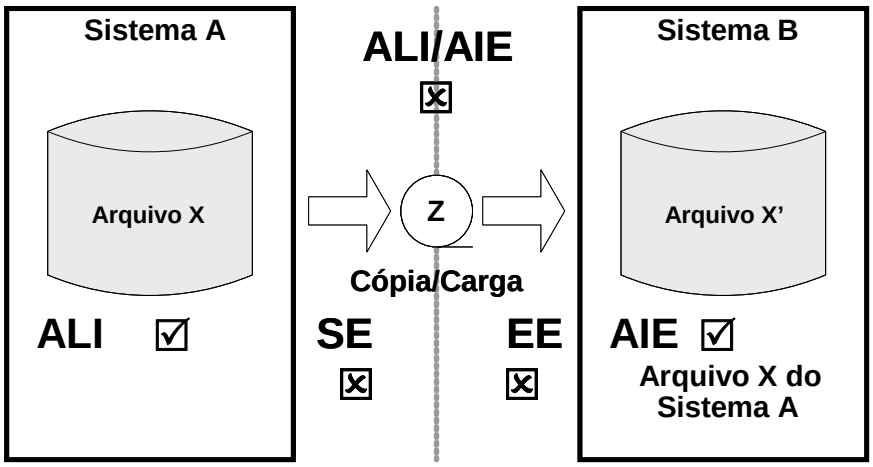
A *intenção primária* é a Aplicação B referenciar os dados que existem logicamente na Aplicação A.

*Transações* – Uma vez que não existem funções lógicas de transação na carga ou propagação da cópia, nenhuma transação é contada em nenhuma das aplicações para suportar a cópia e carga dos dados compartilhados. Então, a Aplicação A não conta a cópia para a Aplicação B como uma SE/CE, e a Aplicação B não a conta como uma EE. Uma indicação adicional seria que o arquivo na Aplicação B é "atualizado" a cada vez com a cópia.

*Arquivos* – Existe apenas um arquivo lógico envolvido. A Aplicação A conta o Arquivo X como um ALI. A Aplicação B conta a tabela copiada do

Arquivo X como um AIE.

**Diagrama da Solução**




**Resumo da Contagem**

| | ALI | AIE | EE | SE/CE | Nota |
|---|---|---|---|---|---|
| Aplicação A | ☑ | | | | |
| Aplicação B | | ☑ | | | |

## Cenário 5: Cópia/*Merge*

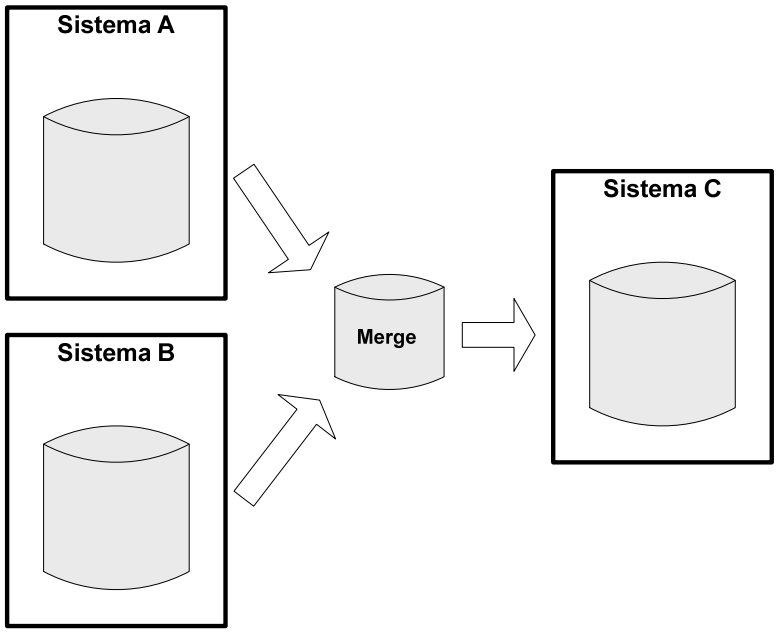
**Descrição**

Dados armazenados em duas aplicações são copiados e fundidos para formar um arquivo que é carregado para dentro de uma terceira aplicação. Diversos arquivos com os mesmos elementos de dados são consolidados em apenas um arquivo.

**Cenário**

Para evitar uma sobrecarga da Aplicação C para procurar dinamicamente os dados das Aplicações A e B, os dados são copiados da Aplicação A e da Aplicação B e fundidos em um novo depósito de dados na Aplicação C. O usuário solicitou que as informações das Aplicações A e B sejam atualizadas diariamente para validação ou apenas para referência. Utilitários de carga, *unload* e *merge* são utilizados. Não existe lógica de processamento envolvida. Isto é normalmente uma solução técnica onde duas aplicações têm diferentes instâncias dos mesmos dados lógicos requeridos por uma terceira aplicação.

**Diagrama**



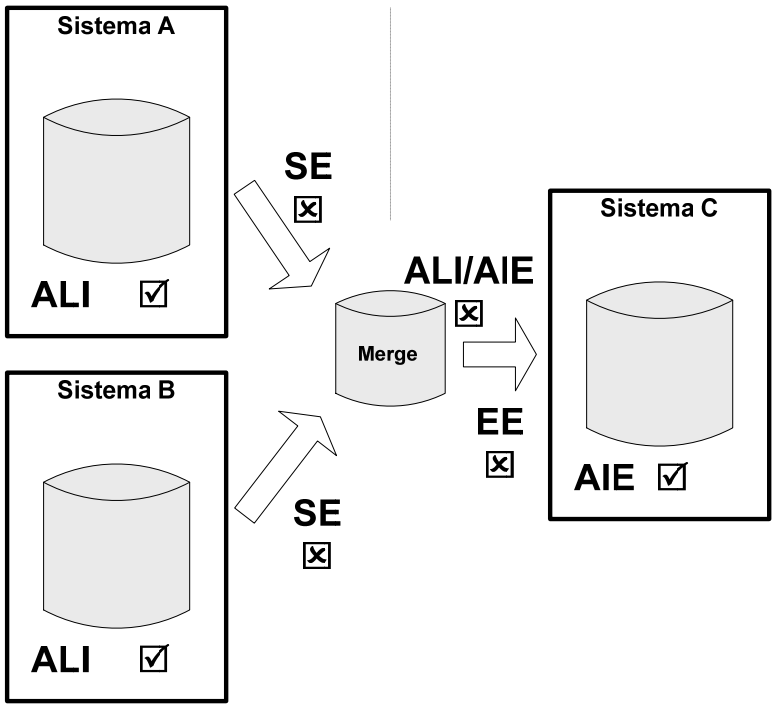

**Interpretação da Contagem**

Logicamente, os depósitos de dados permanecem nas Aplicações A e B. A fusão de dados dentro de um único depósito de dados não é requisito suficiente para a criação um novo ALI para a Aplicação C. Desde que não exista lógica de processamento adicional, nenhuma transação é contada para nenhuma aplicação.

Os dados para a Aplicação C devem ser avaliados de acordo com o item 3.4 da Parte 1 – Medir Funções de Dados e Parte 3, Capítulo 2 – Arquivos Lógicos. Ainda que os dados venham de duas diferentes aplicações, os elementos de dados são exatamente os mesmos (veja definição para

"merge"). Dessa forma, um único arquivo lógico é identificado para a Aplicação C como um AIE.

**Diagrama da Solução**




**Resumo da Contagem**

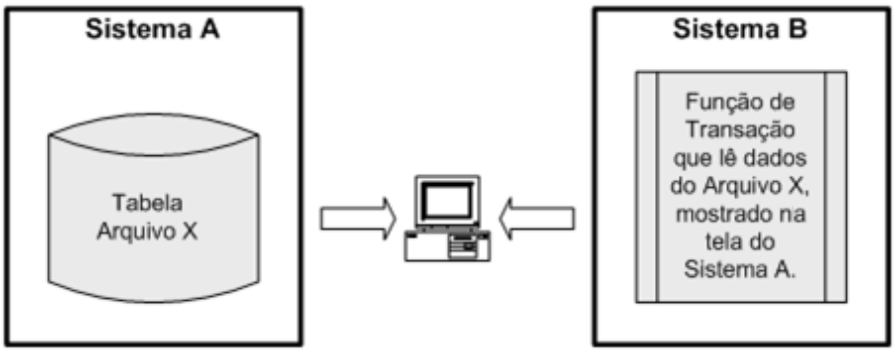
| | ALI | AIE | EE | SE/CE | Nota |
|---|---|---|---|---|---|
| Aplicação A | ☑ | | | | |
| Aplicação B | ☑ | | | | |
| Aplicação C | | ☑ | | | Também contar um ALR na função de transação. |

## Cenário 6: *Screen Scraping*

**Descrição** Acesso a outras transações de telas da aplicação para referenciar/obter dados ou para atualizar aqueles dados da aplicação.

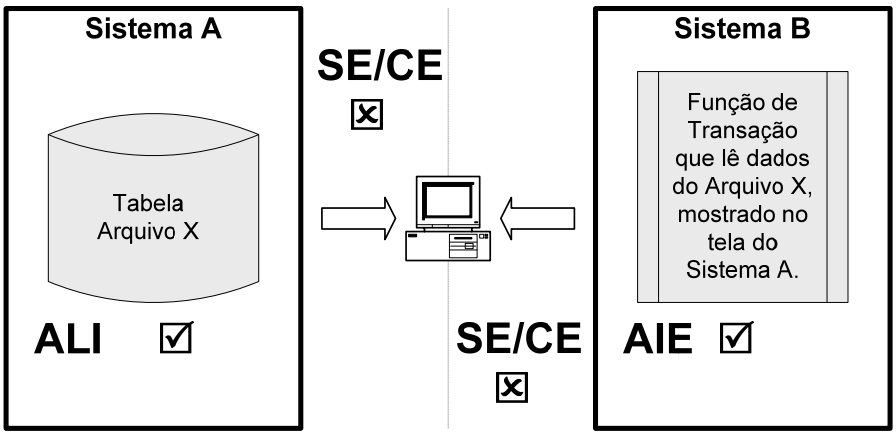
**Cenário** A Aplicação B "lê" o conteúdo de uma tela de consulta na Aplicação A e usa estes dados no processamento de uma função de transação.

**Diagrama**




**Interpretação da Contagem** Logicamente, a Aplicação B está lendo os dados da Aplicação A. A Aplicação A já contou os dados exibidos como um SE/CE (não contados aqui), enquanto a Aplicação B conta os dados como um AIE. Sob uma perspectiva transacional, a Aplicação A é passiva e não conta nada adicionalmente. Para a aplicação B, *screen scraping* é parte do processo elementar da transação e é contado como um ALR (AIE), uma vez que o dado foi originalmente recuperado a partir do ALI da Aplicação A.

**Diagrama da Solução**




**Resumo da Contagem**

| | ALI | AIE | EE | SE/CE | Nota |
|---|---|---|---|---|---|
| Aplicação A | ☑ | | | | |
| Aplicação B | | ☑ | | | |

# Cenários da Contagem – Grupo 2

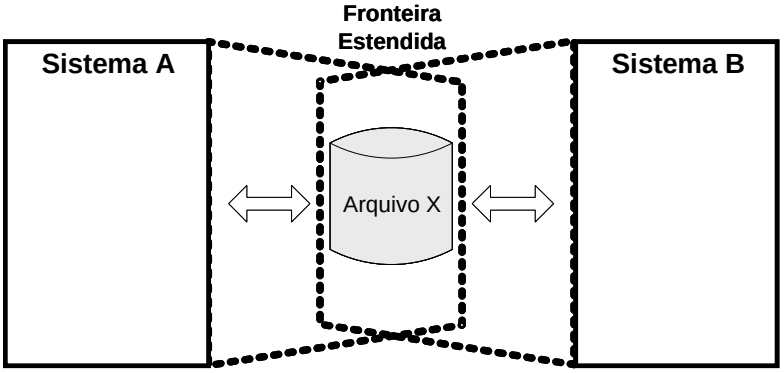
Em cada um dos cenários 7 e 8, a intenção primária é a Aplicação B manter seus próprios dados a partir de dados mantidos pela Aplicação A; isto pode ser implementado como segue:

## Cenário 7: Atualizando o mesmo Depósito de Dados

**Descrição** O mesmo depósito de dados é mantido por duas diferentes aplicações.

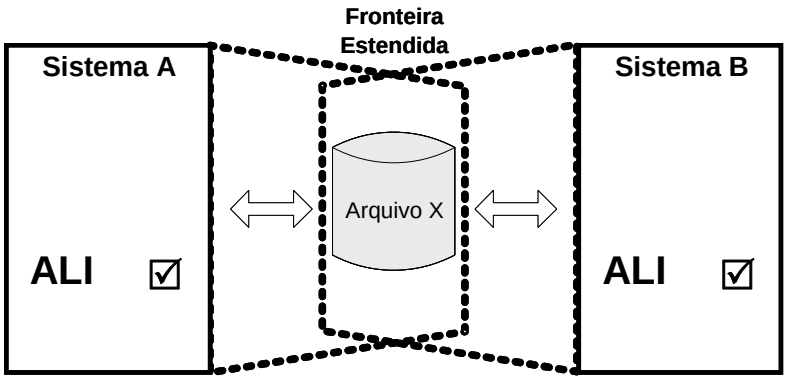
**Cenário** Tanto o Sistema A quanto o Sistema B mantêm o mesmo ALI. Cada um tem sua própria visão dos dados. Existem alguns elementos de dados comuns e outros são únicos para cada sistema.

**Diagrama**




**Interpretação da Contagem** Um ALI é contado para ambas as aplicações, pois cada um tem transações para mantê-lo. As Aplicações A e B mantêm dados no mesmo ALI. Cada aplicação conta apenas um RLR e DERs mantidos, utilizados ou referenciados por aquela Aplicação.

**Diagrama da Solução**




**Resumo da Contagem**

| | ALI | AIE | EE | SE/CE | Nota |
|---|---|---|---|---|---|
| Aplicação A | ☑ | | | | |
| Aplicação B | ☑ | | | | |

## Cenário 8: Dados Padrão de Transação

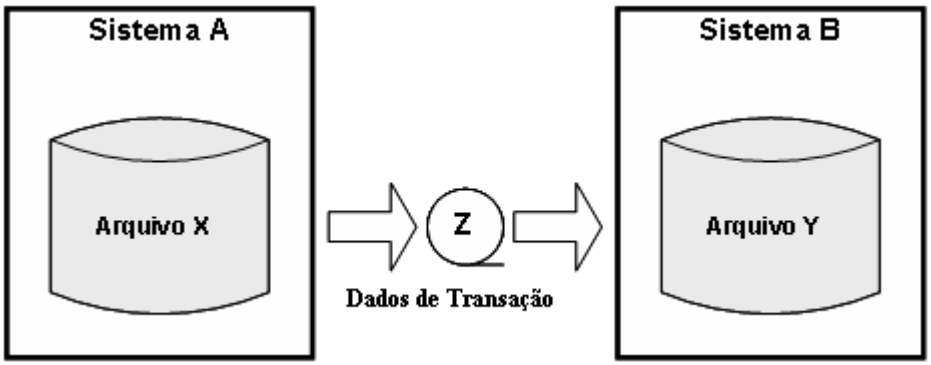
**Descrição** Dados de transação são fornecidos pela aplicação de origem.

**Cenário** A Aplicação A gera um arquivo de transação contendo modificações, o Arquivo Z, que é carregado na Aplicação B. Os registros são geralmente de mais de um tipo. A Aplicação B processa as transações de entrada de acordo com o tipo de transação dos registros do Arquivo Z, antes da atualização dos registros no Arquivo interno Y. Os DERs no Arquivo X da Aplicação A e no Arquivo X' da Aplicação B são diferentes. Por exemplo, o Arquivo X é o Catálogo Principal de Material enquanto o Arquivo Y é uma Lista de Produtos gerada localmente. O processamento inclui os seguintes tipos de transações:

- Inclusão
- Alteração
- Exclusão

Esta transferência de dados é um requisito de negócio do usuário. Tanto a Aplicação A quanto B possuem um requisito para acessar uma versão do Arquivo X, entretanto os DERs nos dois arquivos são diferentes. A Aplicação A envia apenas dados relacionados a alterações. A Aplicação B lê os registros no Arquivo Z e, baseado nos tipos de transação, inicia diferentes lógicas de processamento.

**Diagrama**

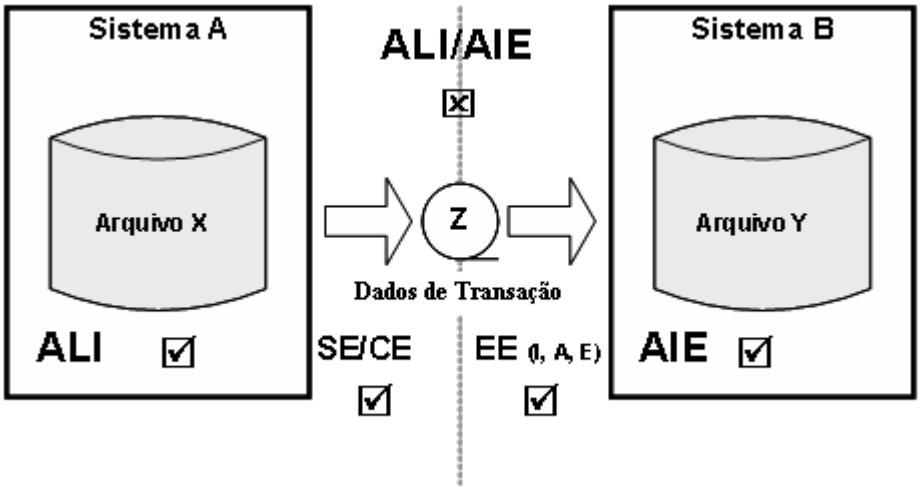



**Interpretação da Contagem** Se cada registro gravado pela Aplicação A no Arquivo Z é processado da mesma forma, apenas uma CE/SE é contada. Apenas quando existirem diferentes lógicas de processamento envolvidas você poderá ter diversas funções de transação (ex., SE/CE) dentro de um único arquivo.

A Aplicação B conta EEs para cada função de manutenção única no Arquivo Y. O número de Tipos de Transações no arquivo de transação Z normalmente determina o número destas funções, mas isto não é necessariamente assim. Diferentes lógicas de processamento devem ser demonstradas.

Existem dois arquivos envolvidos. A Aplicação A conta o Arquivo X como um ALI. A Aplicação B conta o Arquivo Y como um ALI. Nenhum dos dois sistemas conta o Arquivo Z como um arquivo lógico.

**Diagrama da Solução**




**Resumo da Contagem**

| | ALI | AIE | EE | SE/CE | Nota |
|---|---|---|---|---|---|
| Aplicação A | ☑ | | | ☑ | |
| Aplicação B | ☑ | | ☑ | | |

# Resumo

No decorrer deste capítulo, os cenários tiveram foco na utilização de dados dentro da aplicação que está sendo contada e a aplicação das regras de identificação de ALI e AIE relativas à ‘intenção primária’.

Este capítulo não ilustra todas as possíveis implementações de compartilhamento de dados entre aplicações. Entretanto, fornece exemplos suficientes que permitem a um Analista de Pontos de Função aplicar consistentemente as regras de identificação para ALIs e AIEs, focando na intenção primária para a utilização dos dados e fronteiras das aplicações envolvidas na contagem.

# Parte 3 Capítulo 4

# Projetos de Melhoria e Atividades de Manutenção

**Introdução**

Este capítulo fornece diretrizes adicionais na identificação e medição das mudanças funcionais em aplicações instaladas. Não é abordado o relacionamento entre o tamanho funcional da melhoria e o esforço necessário para implementar a melhoria. Para uma visão adicional para esse tópico, o leitor pode consultar a publicação da NESMA, "*Function Point Analysis for Software Enhancement*" [NESMA, 2001], que pode ser obtida através do site www.nesma.org.

Este capítulo também discute as diversas atividades de manutenção e suporte que podem ocorrer durante a vida útil de uma aplicação e qual medida de tamanho funcional fornece uma base útil para estimativas e reconciliação de custo.

**Conteúdo**

Este capítulo inclui as seguintes seções:

# Medindo Projetos de Melhoria

O Tamanho Funcional do Projeto de Melhoria mede as modificações do projeto na aplicação instalada existente que adicionam, modificam ou excluem funções do usuário. Mudanças nas funcionalidades podem ocorrer a partir de novos requisitos, revisão de requisitos do usuário, mudanças legais/regulamentares ou novos usuários.

## Escopo e Fronteira de um Projeto de Melhoria

O Tamanho Funcional do Projeto de Melhoria inclui todas as funções que estão sendo adicionadas, alteradas e excluídas. A(s) fronteira(s) da(s) aplicação(ões) impactada(s) permanece(m) a(s) mesma(s). As funcionalidades da(s) aplicação(ões) refletem o impacto das funções sendo adicionadas, alteradas ou excluídas.

Pode existir mais de uma aplicação incluída no escopo da contagem. Dessa forma diversas fronteiras deverão ser identificadas, resultando em um tamanho funcional do projeto de melhoria separado para cada aplicação afetada.

Se o tamanho total do projeto de melhoria é requerido, ele é calculado pela soma total das contagens de melhoria para todas as aplicações incluídas no escopo da contagem.

## Medindo Funções de Dados em Projetos de Melhoria

A inclusão de novos arquivos lógicos internos ou arquivos de interface externa em um projeto de melhoria normalmente são facilmente identificados e medidos de acordo com as regras definidas na Parte 1. Capítulo 6 da Parte 2 (Medindo Funções de Dados) e Capítulo 2 da Parte 3 (Arquivos Lógicos) contem orientação adicional para medição de funções de dados bem como definições dos termos relacionados.

Porém, considerações podem ser levantadas, como a seguir:

- Se a mudança envolve apenas a inclusão de novos registros no arquivo lógico ou novos valores em um atributo existente dentro do arquivo lógico, não existe justificativa para contar a função de dado como sendo alterada.
- Se uma função de dado é alterada porque um atributo está sendo incluído e este atributo não é utilizado pela aplicação que está sendo medida, então não existe mudança naquela aplicação.

- Para que uma função de dado seja contada como uma função alterada, é obrigatório que a função seja estruturalmente alterada (ex.: inclusão ou remoção de atributos ou alteração de características de um atributo).

  **Nota**: Um novo texto de ajuda é frequentemente adicionado à função de dados "Ajuda" que auxilia a nova transação. Uma vez que não há mudança na estrutura da função de dados "Ajuda", não será contada como alterada.

- Se uma aplicação está solicitando o uso (para referenciar ou manter) um atributo existente que não era utilizado antes, então a função de dado relacionada é considerada alterada para aquela aplicação. Isto pode ocorrer sem que ocorra nenhuma mudança física no arquivo.
- Se novos atributos são incluídos em um ALI, procure por funções de transação novas ou modificadas que mantém o atributo neste ALI para confirmar que a mudança ocorreu.
- Se um atributo é incluído em um ALI que é mantido por duas aplicações e se uma das aplicações mantém o novo atributo, mas a outra apenas referencia este atributo, então ambas as aplicações consideram o ALI alterado. Entretanto, a segunda aplicação não terá nenhuma função de transação nova ou alterada que mantenha este campo naquele ALI.
- Se uma aplicação não mantém nem referencia um atributo novo ou alterado, então esta aplicação não pode considerar esta função de dados como alterada.
- Se um arquivo físico é incluído por um projeto de melhoria, ele não resulta necessariamente em um novo arquivo lógico. Primeiramente precisamos determinar se o novo arquivo físico é uma mudança de um arquivo lógico existente com DERs adicionais e possivelmente um novo RLR, ou um novo arquivo lógico. Orientações adicionais em relação à medição de arquivos lógicos podem ser obtidas no Capítulo 6 da Parte 2 e no Capítulo 2 da Parte 3.

## Medindo Funções de Transação em Projetos de Melhoria

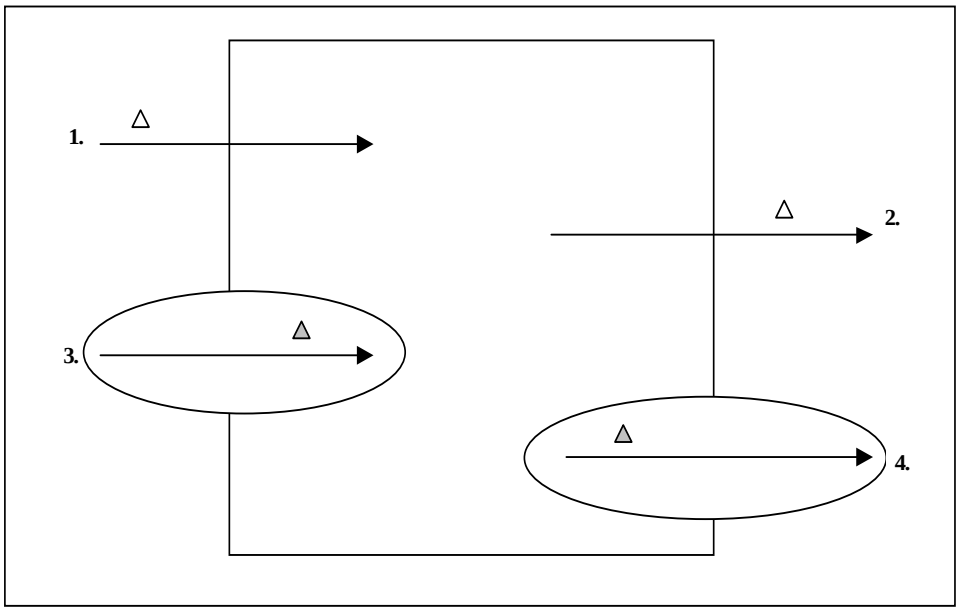
A inclusão de novas funções de transação geralmente são facilmente identificadas e medidas de acordo com as regras definidas na Parte 1. Orientações adicionais em relação à medição de funções de transação bem como definições de termos relacionados podem ser obtidas no Capítulo 7 da Parte 2 (Medindo Funções de Transação).

A identificação das funções de transação que foram alteradas pela inclusão ou exclusão de elementos de dados (DERs) é óbvia. Entretanto, não é óbvio quando os requisitos do usuário são para mudanças na lógica de processamento, como descrito na Parte 1. Quando a lógica de processamento sofreu alteração dentro da aplicação para satisfazer requisitos de negócio, o

processo elementar que incorpora aquela lógica deve ser identificado e contado como sendo alterado.

Uma simples mudança na lógica de processamento nem sempre afeta todas as transações relacionadas.

Por exemplo, quando uma alteração de edição ou validação é feita na lógica de processamento de entrada de dados e as transações existentes são de Inclusão, Alteração, Exclusão e Consulta Implícita, então apenas as transações de Inclusão e Alteração são contadas para a melhoria. A menos que exista uma alteração específica na lógica de exclusão (ex.: edição de integridade referencial) ou alteração na lógica de consulta (ex.: seleção ou recuperação), as transações de Exclusão e Consulta Implícita não serão alteradas.




No diagrama acima:

1. A transação 1 é modificada porque um DER adicional foi incluído e está cruzando a fronteira.
2. Igualmente, a transação 2 onde um DER adicional está cruzando a fronteira será contada como alterada.
3. Para a transação 3, uma rotina interna de validação da aplicação sofreu mudança. Uma vez que esta é uma mudança na lógica de processamento em um processo elementar, a transação 3 associada é contada como sendo alterada.
4. Para a transação 4 onde um critério de seleção ou um filtro sofreu mudança, a transação é contada como uma funcionalidade modificada.

Em alguns casos, uma mudança específica pode afetar como várias funções de transação são processadas. Sem considerar se as mudanças na lógica foram feitas fisicamente em uma rotina comum utilizada por várias transações, as

funções alteradas devem ser identificadas baseadas nos processos elementares que incorporam aquela lógica. Se vários processos elementares são afetados, então conte diversas funções de transação. Se apenas um único processo elementar foi afetado, conte uma função de transação como sendo alterada. Em todos os casos, os requisitos do usuário e a visão do negócio devem ser fatores determinantes.

Por exemplo, o requisito é para modificar a edição em "Novos Pedidos" para incluir uma validação para o saldo devedor do cliente. Se existirem diversas transações de "Pedido" diferentes (individual, comercial, administrativo, etc.), mas a edição alterada é apenas na transação de "Pedido Comercial", então apenas esta transação deve ser contada como alterada. Mas se o requisito é para modificar todos os tipos de pedido, então cada transação separadamente deve ser contada como alterada.

Outra indicação de que uma função de transação deve ser contada seria a cobertura de casos de testes. Um único grupo de casos de testes indicaria que um único processo elementar foi alterado.

Freqüentemente a natureza de uma mudança é relatada pelo desenvolvedor, declarando que o único módulo que está sendo alterado é usado na produção de um grande número de relatórios ou extrações, ou no processamento de muitas transações de entrada. Todas as funções podem usar a rotina em comum, mas apenas um grupo dessas funções usa a edição que está sendo alterada nessa rotina em comum. Apenas as funções que incorporam a lógica de processamento alterada devem ser contadas como alteradas. O desafio reside na avaliação adequada do nível apropriado da mudança funcional. A ênfase deve ser nos requisitos de negócio, com o tamanho funcional do projeto de melhoria refletindo a intenção dos requisitos do usuário.

## Lógica de Processamento

*Lógica de processamento* é definida como qualquer um dos requisitos especificamente solicitados pelo usuário para completar um processo elementar como validações, algoritmos ou cálculos e leitura ou manutenção de uma função de dados. Estes requisitos podem incluir as seguintes ações:

1. Validações são efetuadas

   Por exemplo, na inclusão de um novo funcionário em uma organização, o processo de funcionário valida o DER empregado.

   - Se um requisito existe para executar uma validação diferente ou alterar a validação em uma função de transação existente, a transação, com o tamanho funcional alterado seria contada no projeto de melhoria.

2. Formulas matemáticas e cálculos são executados

   Por exemplo, quando relacionar todos os funcionários de uma organização, o processo inclui o cálculo do número total de funcionários assalariados, funcionários horistas e de todos os funcionários.

   - Se um requisito de negócio existe para modificar um cálculo existente (ex.: antes a fórmula era A + B = C e agora será C = A * B), a função que inclui esse cálculo seria contada com o tamanho funcional alterado no projeto de melhoria.
   - Atualmente existe uma lista de funcionários que é contada como uma CE. O requisito do projeto de melhoria determina mostrar um resumo da contagem de todos os funcionários. A transação deve ser identificada como alterada e o tipo de função deve ser alterado de uma CE para uma SE no projeto de melhoria.

3. Valores equivalentes são convertidos

   Por exemplo, a idade do empregado é convertido para uma faixa etária usando uma tabela.

   - Se um requisito de negócio existe para alterar uma funcionalidade para incluir a habilidade de converter o salário do empregado em uma faixa salarial, a função seria contada com o tamanho funcional alterado no projeto de melhoria.

4. Dados são filtrados e selecionados pela utilização dos critérios especificados para comparar vários grupos de dados

   Por exemplo, para gerar uma lista de funcionários por sua função, um processo elementar compara o número da função ao de uma função atribuída para o funcionário, para selecionar e relacionar os funcionários assinalados a estas funções.

   - Se um requisito existe para modificar o critério de seleção ou incluir um critério de seleção adicional, excluindo alterando ou incluindo valores, para uma transação existente (uma lista de funcionários agora precisa mostrar uma lista de funcionários que tenham sido nomeados para um cargo em menos de um ano), a transação, com o tamanho funcional alterado seria contada no projeto de melhoria.
   - Se um requisito existe apenas para alterar o(s) valor(es) de um critério existente, como selecionar um departamento diferente ou adicionar mais um departamento na lista de departamentos, então não tem contagem para a transação.
   - Se o requisito é para alterar o critério de seleção de um único departamento para uma lista de departamentos, isso seria contato com uma mudança.

- Se um requisito existe para alterar a tela de pesquisa de funcionários para adicionar um filtro no local, esse filtro adicional não cria um novo processo elementar. Essa função é contada com o tamanho funcional alterado no projeto de melhoria.

5. Condições são analisadas para determinar quais são aplicáveis

   Por exemplo, a lógica de processamento é empregada por um processo elementar quando um funcionário é incluído dependerá se um funcionário é pago baseado em seu salário ou nas horas trabalhadas. A entrada dos DERs (e o resultado do processamento lógico) baseado em uma escolha diferente (salário ou horas trabalhadas) nesse exemplo é parte de um processo elementar.

   - Se um requisito existe para modificar a condição ou incluir condições adicionais em uma transação existente, a transação, com o tamanho funcional alterado seria contada no projeto de melhoria.

6. Um ou mais ALIs são atualizados

   Por exemplo, quando incluir um funcionário, o processo elementar atualiza o ALI funcionário para manter os dados do funcionário.

   - Se um requisito de negócio resulta na atualização de um ALI adicional ou diferentes DERs pela transação existente, a transação, com o tamanho funcional alterado seria contada no projeto de melhoria.

7. Um ou mais ALIs ou AIEs são referenciados

   Por exemplo, ao incluir um funcionário, o AIE "moeda" é referenciado para determinar o valor da hora do funcionário com a correta taxa de conversão para dólar.

   - Se um requisito de negócio referencia novos ALIs, AIEs ou DERs em uma transação existente, a transação afetada seria contada com o tamanho funcional alterado no projeto de melhoria.

8. Dados ou informações de controle são recuperados

   Por exemplo, para visualizar uma lista de empregados, as informações do empregado são recuperadas de uma função de dados.

   - Se um requisito de negócio estabelece a recuperação de informações adicionais em uma transação existente, a transação afetada seria contada com o tamanho funcional alterado no projeto de melhoria.

9. Dados derivados são criados pela transformação de dados existentes para criação de dados adicionais

   Por exemplo, para determinar (derivar) um número de registro do paciente (ex. SILJO01), o seguinte dado é concatenado:

   - as primeiras 3 letras do último nome do paciente (ex., SIL para Silva)
   - as primeiras 2 letras do primeiro nome do paciente (ex., JO para João)
   - um número seqüencial de dois dígitos (começando de 01)

- Se o requisito de negócio resulta na mudança em como a transação deriva os dados, a transação afetada seria contada com o tamanho funcional alterado no projeto de melhoria.

10. O comportamento da aplicação é alterado

    Por exemplo, o comportamento do processo elementar de pagamento de funcionários é alterado quando uma mudança é feita para pagamentos toda a sexta-feira ao invés de pagamentos realizados no 15o. dia e no último dia do mês; resultando em 26 períodos de pagamento por ano ao invés de 24.

- Se o requisito de negócio resulta na alteração do comportamento do sistema (ex.: no exemplo acima, a transação é alterada para que o parâmetro data de pagamento afete apenas os funcionários horistas e não a todos os funcionários), a transação afetada seria contada com o tamanho funcional alterado no projeto de melhoria.

11. Preparar e apresentar informações para fora da fronteira

    Por exemplo, uma lista de funcionários é formatada e exibida para o usuário.

- Quando o requisito de negócio resulta na apresentação de DERs adicionais para fora da fronteira, a transação afetada seria contada com o tamanho funcional alterado no projeto de melhoria.
- Mudanças em literais, formatos, cores e outros elementos da apresentação física não são considerados mudanças na lógica de processamento e portanto não fazem parte do tamanho funcional do projeto de melhoria.
- Quando o requisito de negócio é enviar um arquivo de saída existente para uma aplicação diferente ou uma nova aplicação sem alterar de nenhuma forma o processamento lógico (ex.: critério de seleção, cálculos), não afeta o tamanho funcional do projeto de melhoria.
- Rearranjar dados na tela, relatório ou arquivo exibindo elementos de dados existentes em uma nova posição não é considerado alteração na

lógica de processamento e não é contada como uma melhoria.

- Alterações nas características (ex.: tamanho, tipo, precisão, etc.) de um atributo cruzando a fronteira que deve ter uma mudança para outra forma de lógica de processamento (ex.: validações, cálculos) a ser contada.
- Quando uma tela nova ou alterada requer uma ajuda adicional ou altera uma ajuda existente, a mudança na função Help não é contada porque é apenas inclusão ou atualização de texto ou valores.

12. Existe a capacidade de receber dados ou informações de controle que entram pela fronteira da aplicação

    Por exemplo, um usuário entra com informações para adicionar um pedido do cliente para a aplicação.

    - Quando o requisito de negócio resulta em diferentes DERs que entram pela fronteira, a transação afetada seria contada com o tamanho funcional alterado no projeto de melhoria.
    - Quando o requisito de negócio é para aceitar a entrada de um arquivo existente de uma aplicação adicional ou diferente sem mudanças em nenhuma lógica de processamento (ex.: validações, cálculos), não há impacto no tamanho funcional do projeto de melhoria.
    - Mudanças nas características (ex.: tamanho, tipo, precisão, etc.) de um atributo cruzando a fronteira deve ter mudanças para outra forma de lógica de processamento (ex.: validações, cálculos) para ser contado.

13. Classificando ou organizando um grupo de dados. Essa forma de lógica de processamento não impacta a identificação do tipo ou contribuição da singularidade do processo elementar; ou seja, a organização dos dados não constitui uma singularidade.

    Por exemplo, lista de empregados é classificada tanto por ordem alfabética quanto por ordem de local.

    Por exemplo, em uma tela de entrada de pedido, o cabeçalho com informações do pedido é exibido na parte de cima da tela, e os detalhes do pedido são exibidos abaixo.

    **Nota:** Alterações na seqüência de classificação normalmente são contadas. Alterações na organização por si só não são normalmente contadas.

    - Quando o requisito de negócio resulta na mudança da seqüência de reclassificação existente (ex.: o usuário solicita a lista de funcionários acima referenciada em ordem de localização, ao invés de em ordem alfabética), a transação afetada seria contada com o tamanho funcional alterado no projeto de melhoria.

- Em uma tela de entrada de pedidos, o usuário solicita que as informações do cabeçalho do pedido sejam colocadas à esquerda das informações do detalhe do pedido ao invés de acima. Não há contagem para essa mudança.
- O usuário solicita que o atributo Sobrenome na tela de Contratação de Empregado seja exibido a esquerda da inicial do nome do meio e do primeiro nome. Não há contagem para esse reposicionamento. Se o requisito é também para preencher os dados conforme o ultimo sobrenome digitado, então há uma mudança na lógica de processamento e a função de Contratação de Usuário é contada.
- O usuário solicita um relatório adicional com os mesmos dados (lista de funcionários) classificados pela localização. Uma nova transação não deve ser contada, mas uma mudança na função existente seria incluída no tamanho funcional do projeto de melhoria.

Um processo elementar pode incluir diversas alternativas ou ocorrências das ações acima. Por exemplo: validações, filtros, reclassificações, etc.

# Considerações e Dicas

Muitos projetos de melhoria envolvem mudanças apenas na lógica de processamento sem mudanças físicas de entradas, saídas ou arquivos (ex.: atributos incluídos ou excluídos). A seguir estão itens ou questões que podem ser discutidas com os desenvolvedores durante a medição de melhoria para aplicações instaladas:

## Questões a Considerar num Projeto de Melhoria

As questões listadas abaixo podem ser usadas durante as entrevistas com os desenvolvedores ou especialistas de negócio durante as sessões de medição do tamanho funcional. Respostas positivas nas questões listadas abaixo indicam a possibilidade de mudanças na funcionalidade do usuário. Investigação adicional é necessária para determinar se e como elas afetariam o tamanho funcional.

**Quais funcionalidades NOVAS foram criadas aos usuários na aplicação?**

- Existe algum novo arquivo permanente do usuário, base de dados, tabelas, entidades ou objetos que foram desenvolvidos/criados neste projeto?
- A aplicação agora está recebendo e processando novas transações de entrada ou arquivos de entrada que não estavam antes em produção?
- Existe alguma nova tela sendo construída para o usuário?
- Existem novas interações (interfaces) com outras aplicações?
- A aplicação está gerando novas saídas, relatórios ou arquivos para outros sistemas?
- Existe algum tipo de registro novo que está sendo incluído em um arquivo de transação existente?
- Existe algum novo processo de consulta estabelecido pelo usuário?
- Existe algum processo batch sendo incluído?

**Quais funcionalidades do usuário foram ALTERADAS/MODIFICADAS para atender aos requisitos do usuário?**

- Existe algum arquivo permanente, base de dados ou tabelas existentes que tenham atributos ou colunas incluídas ou excluídas? Existem características de qualquer atributos ou coluna existente sendo modificados? Não deve ser contada a inclusão de novos valores ou novas linhas em atributos existentes de tabelas existentes.
- Existe alguma lógica de processamento sendo modificada, existente em telas de entradas ou em entradas recebidas via arquivos enviados pelos

usuários ou por outras aplicações ?

- Foi efetuada alguma alteração nas atuais telas de entrada ou em arquivos de transação vindos de outra aplicação (ex.: novos atributos, mudanças nos atributos dos atributos existentes) ?
- Existe algum relatório do usuário, arquivos de saída ou telas existentes sendo modificadas pela inclusão ou exclusão de atributos?
- Existe algum relatório do usuário, arquivos de saída ou telas existentes sendo modificadas pela inclusão ou exclusão de campos?
- Para as saídas existentes, existe alguma alteração na lógica de processamento da geração destes arquivos, relatórios ou telas?
- Existe alguma tela de consulta sendo modificada?
- Foi feita alguma alteração na edição, nos critérios de seleção ou nos filtros associados às telas de consulta?
- Existe algum processo batch sendo alterado?

**Quais funcionalidades do usuário foram DELETADAS/EXCLUÍDAS devido a requisitos do usuário?**

- Quais funcionalidades do usuário foram excluídas da produção?
- Existe algum arquivo permanente que não é mais necessário?
- A aplicação parou de aceitar algum arquivo de entrada de outro sistema ou excluiu alguma tela antes acessada pelo usuário?
- Existe algum relatório sendo removido porque o usuário não precisa mais dele?
- Existe algum arquivo de saída para outro sistema sendo eliminado?
- Existe alguma consulta do usuário sendo excluída?
- Existe algum processo batch sendo excluído?

**Quais Funcionalidades de Conversão estão sendo fornecidas?**

- Existe algum processamento de dados executado uma única vez a ser criado para limpar dados, organizar ou popular novos atributos como resultado de modificações na estrutura em arquivos permanentes?
- Existe algum requisito para popular algum novo arquivo permanente?
- Existe alguma solicitação do usuário para conversão de relatórios?

**Quais outras mudanças estão sendo feitas na aplicação para este projeto?**

- Esta pergunta pode incentivar o especialista de negócios ou o pessoal de desenvolvimento através do fornecimento de dicas adicionais para as mudanças que podem ser contadas.

## Resumo das Considerações e Dicas

Estas diretrizes auxiliam na determinação das funcionalidades entregues pelo projeto e seu impacto no *baseline* da aplicação que está sendo medida. A combinação de novas funcionalidades incluídas, o efeito das mudanças feitas nas funcionalidades existentes e as funcionalidades excluídas devem ser utilizadas para medir o tamanho do Projeto de Melhoria, assim como para atualizar o *baseline* de Pontos de Função da aplicação..

# Informações para Medição de Projetos de Melhoria

Além da documentação identificada na Parte 2 Capítulo 3 Coletar Documentação Disponível, os seguintes itens também devem ser fornecidos para um projeto de melhoria:

- Documentação da Medição do Tamanho Funcional da Aplicação existente
- Requisitos do usuário de qualquer alteração/modificação da aplicação existente
- Layout das telas, relatórios e/ou diagrama de fluxo de dados das funções batch existentes para todas as funções afetadas, mostrando como elas eram ANTES do projeto de melhoria
- Layouts revisados das telas, relatórios e/ou diagrama do fluxo de dados das funções batch de todas as funções afetadas, mostrando como elas ficarão APÓS o projeto de melhoria
- Documentação de todas novas funcionalidades (novos relatórios, saídas, entradas de dados ou entidades) que serão incluídas na aplicação como parte do projeto de melhoria
- Layouts das telas, relatórios e/ou diagrama do fluxo de dados das funções batch que serão EXCLUÍDAS da aplicação como resultado do projeto de melhoria.

É reconhecido que, em muitos casos, a documentação listada acima pode não estar disponível ou não ser aplicável. A fonte de informação mais importante é o conhecimento do especialista da aplicação e a documentação dos requisitos.

Se o baseline de uma aplicação não existe, um cuidado deve ser tomado no estabelecimento da fronteira e na identificação das funções de dados e de transação.

# Procedimentos

Os passos a seguir são sugeridos para medir o tamanho funcional de um projeto de melhoria; no entanto, eles podem ser executados em qualquer ordem.

| Passo | Ação |
|---|---|
| **1** | Reúna e revise a documentação disponível. |
| **2** | Converse com o especialista de negócios para discutir as mudanças planejadas/executadas. |
| **3** | Identifique e avalie as funcionalidades incluídas. |
| **4** | Identifique e avalie as funcionalidades alteradas:<br>• Determine as complexidades das funções antes da alteração (a partir da documentação da última medição do tamanho funcional, ou meça como existiam antes da mudança).<br>• Determine as complexidades das funções depois da alteração. |
| **5** | Identifique e avalie as funcionalidades excluídas. |
| **6** | Identifique e avalie qualquer funcionalidade de conversão ou de execução uma única vez, solicitada para implementação desta melhoria. |

# Exemplo de Projetos de Melhoria

**Exemplo**

Esta seção mostra um exemplo de um projeto de melhoria. Os requisitos para o projeto de melhoria incluem as seguintes mudanças:

- O usuário precisa receber um relatório adicional sobre as funções que incluem totais.
- DERs adicionais são requeridos durante a inclusão de funções no modo batch e na correção de transações suspensas. Uma referência na segurança também é incluída na transação para inclusão de função.
- A função de Atribuição de Função deve verificar que tipo de função atribuída a um funcionário se iguala com a classificação do funcionário, que é mantido no sistema de Relações do Funcionário.
- O relatório de Funcionários por tempo na função deve ser alterado. Ao invés de exibir uma contagem de funcionários acima de 12 e 24 meses, o relatório deve agora mostrar apenas os funcionários assalariados.
- O usuário não precisa mais incluir uma função on-line; portanto esta funcionalidade deve ser ou foi excluída.

**Funcionalidade da Aplicação**

Os seguintes parágrafos explicam as funcionalidades da aplicação medidas para o exemplo de projeto de melhoria. As funcionalidades são descritas como incluídas, alteradas ou excluídas.

**Funcionalidades Incluídas**

A seguinte tabela mostra a complexidade funcional para as funcionalidades incluídas medidas quando o projeto foi entregue.

**Nota 1:** O fornecimento de um novo relatório é uma saída externa adicional.

**Nota 2:** Os dados do Sistema de Relações do Funcionário (RF) referenciado pela aplicação de RH é identificado como um AIE.

| Funções de Dados | RLRs | DERs | Complexidade |
|---|---|---|---|
| **Arquivo de Interface Externa** | | | |
| Dados do Sistema RF | 1 | 2 | Baixa |

| Funções de Transação | ALRs | DERs | Complexidade |
|---|---|---|---|
| **Saída Externa** | | | |
| Relatório da Função | 1 | 15 | Baixa |

## Funcionalidades Alteradas

A seguinte tabela mostra a complexidade funcional para as funcionalidades alteradas e as funções ficarão depois que o projeto de melhoria estiver pronto.

**Nota 1:** A complexidade para a inclusão de uma função foi aumentada porque um novo tipo de arquivo é referenciado. A complexidade para a correção de transações suspensas permanece baixa.

**Nota 2:** Mesmo que um Tipo de Arquivo Referenciado adicional seja contado, não existe mudança na complexidade para a inclusão da atribuição da função pois ela já é Complexa.

**Nota 3:** Embora exista uma mudança no critério de seleção e na lógica de soma, não existe mudança na complexidade do relatório.

| Funções de Transação | ALRs | DERs | Complexidade |
|---|---|---|---|
| **Entrada Externa** | | | |
| Inclusão das informações sobre funções (batch) | 3 | 8 | Alta |
| Correção de Transação Suspensa | 1 | 8 | Baixa |
| Inclusão da Atribuição da Função | 4 | 7 | Alta |
| **Saída Externa** | | | |
| Tempo dos Funcionários na Atribuição | 3 | 7 | Média |

## Funcionalidades Excluídas

A seguinte tabela mostra a complexidade funcional para as funcionalidades excluídas identificadas no final do projeto.

| Funções de Transação | ALRs | DERs | Complexidade |
|---|---|---|---|
| **Saída Externa** | | | |
| Inclusão das informações sobre funções (tela) | 1 | 7 | Baixa |

## Contribuição da Aplicação ao Tamanho Funcional

Os parágrafos a seguir explicam a contribuição da funcionalidade de aplicação para o tamanho funcional total.

### Funcionalidades Adicionadas

A tabela a seguir mostra a contribuição para o tamanho funcional para as funcionalidades adicionadas identificadas no final do projeto.

| Tipo da Função | | Complexidade Funcional | | Totais por Complexidade | Totais por Tipo de Função |
|---|---|---|---|---|---|
| AIE | 1 | Baixa | X 5 = | 5 | |
| | | Média | X 7 = | | |
| | | Alta | X 10 = | | |
| | | | | | 5 |
| SE | 1 | Baixa | X 4 = | 4 | |
| | | Média | X 5 = | | |
| | | Alta | X 7 = | | |
| | | | | | 4 |

### Funcionalidades Alteradas

A tabela a seguir mostra a contribuição para o tamanho funcional das funcionalidades alteradas como passarão a existir após o projeto de melhoria ser concluído.

| Tipo da Função | | Complexidade Funcional | | Totais por Complexidade | Totais por Tipo de Função |
|---|---|---|---|---|---|
| EE | 1 | Baixa | X 3 = | 3 | |
| | | Média | X 4 = | | |
| | 2 | Alta | X 6 = | 12 | |
| | | | | | 15 |
| SE | | Baixa | X 4 = | | |
| | 1 | Média | X 5 = | 5 | |
| | | Alta | X 7 = | | |
| | | | | | 5 |

### Funcionalidades Excluídas

A tabela a seguir mostra a contribuição para o tamanho funcional das funcionalidades excluídas.

| Tipo da Função | | Complexidade Funcional | | Totais por Complexidade | Totais por Tipo de Função |
|---|---|---|---|---|---|
| EE | 1 | Baixa | X 3 = | 3 | |
| | | Média | X 4 = | | |
| | | Alta | X 6 = | | |
| | | | | | 3 |

## Cálculo Final

Usando a complexidade e contribuições desse exemplo, o tamanho funcional do projeto de melhoria é exibido abaixo.

EFP = ADD + CHGA + CFP + DEL

Onde

- EFP é a contagem de pontos de função do projeto de melhoria
- ADD é o tamanho das funções que estão sendo adicionadas pelo projeto de melhoria
- CHGA é o tamanho das funções sendo alteradas pelo projeto de melhoria – como elas são / serão após a implementação
- CFP é o tamanho das funcionalidades de conversão
- DEL é o tamanho das funções sendo excluídas pelo projeto de melhoria

EFP = 9 + 20 + 0 + 3

EFP = 32

## Tamanho Funcional Inicial da Aplicação

O tamanho funcional inicial da aplicação é exibido abaixo.

AFP = ADD

Onde

- AFP é a contagem de pontos de função da aplicação
- ADD é o tamanho das funções que serão entregues para o usuário pelo projeto de desenvolvimento (excluindo o tamanho de qualquer funcionalidade de conversão) ou a funcionalidade que sempre existiu quando a aplicação foi medida

AFP = 115

**Nota:** Apenas o tamanho das funcionalidades da aplicação instaladas para o usuário são incluídas no tamanho funcional inicial da aplicação.

## Tamanho Funcional da Aplicação Após a Melhoria

A medição do tamanho funcional da aplicação para refletir as melhorias é exibida abaixo.

AFPA = (AFPB + ADD + CHGA) - (CHGB + DEL)

onde

- AFPA é a contagem de pontos de função após o projeto de melhoria
- AFPB é a contagem de pontos de função antes o projeto de melhoria
- ADD é o tamanho das funções sendo adicionadas pelo projeto de melhoria
- CHGA é o tamanho das funções sendo alteradas pelo projeto de melhoria – como elas são / serão após a implementação
- CHGB é o tamanho das funções sendo alteradas pelo projeto de melhoria – como elas são / eram antes da implementação
- DEL é o tamanho das funções sendo excluídas pelo projeto de melhoria

AFPA = (115 + 9 + 20) - (18 + 3)

AFPA = 123

Algumas pessoas podem utilizar um valor de fator de ajuste (VAF), que considera as 14 características gerais do sistema (GSCs). Para orientações para utilizar do VAF e dos GSCs, consulte o Apêndice C.

# Considerações sobre Melhorias e Manutenções

Uma vez que uma aplicação foi desenvolvida e instalada, ela deve ser mantida (modificada) a fim de continuar satisfazendo às constantes necessidades de mudanças do negócio e do ambiente técnico. Esta manutenção inclui um grupo de atividades que são executadas durante esta fase do ciclo de vida da aplicação, algumas delas envolvem mudanças funcionais que se aplicam a APF.

## Categorias de Manutenção

O IEEE (Institute of Electrical and Electronics Engineers Inc.) define três categorias de manutenção:

**Manutenção Adaptativa:** Manutenção para fazer com que o software continue sendo utilizável em um ambiente alterado.

**Manutenção Corretiva:** Manutenção para corrigir falhas no hardware ou software.

**Manutenção Perfectiva:** Manutenção para melhorar a performance, facilidade de manutenção ou outros atributos do software instalado.

Enquanto este capítulo fornece dicas e diretrizes para contagem de Pontos de Função para melhorias em aplicações existentes, não existem padrões na indústria para classificação consistente de atividades que se enquadram nas categorias acima. Esta seção fornece uma estrutura de trabalho baseada na experiência comum da indústria através da qual se pode avaliar a aplicabilidade de APF no suporte a aplicações instaladas.

A ISO (International Organization for Standardization) e IEC (International Electrotechnical Commission) definem três categorias de manutenção:

**Manutenção Adaptativa** A modificação de um sistema, realizada após a entrega, para manter um software utilizável em um ambiente alterado ou em alteração. Manutenção adaptativa fornece as melhorias necessárias para adaptar as modificações no ambiente em que o software deve funcionar. Essas mudanças são aquelas que devem ser realizadas para regular com o ambiente em alteração. Por exemplo, o sistema operacional deve ser atualizado e algumas alterações podem ser feitas para adaptar o novo sistema operacional. (ISO/IEC 14764:2006)

**Manutenção Corretiva** A modificação reativa de um software realizada depois da entrega para corrigir problemas descobertos. A modificação corrige o software para satisfazer requisitos. (ISO/IEC 14764:2006)

**Manutenção Perfectiva**

Manutenção de um software após a entrega para detectar e corrigir falhas ocultas no software antes que elas se manifestem como falhas. Manutenção perfectiva fornece melhorias para o usuário, melhoria da documentação do programa, e recodificação para melhorar a performance, manutenção ou outros atributos do software. Contraste com: manutenção adaptativa; manutenção corretiva. (ISO/IEC 14764:2006)

Medição do tamanho funcional quantifica o tamanho dos requisitos de negócio. Em um ambiente de melhoria, mede os efeitos dessas mudanças nos requisitos de negócio. Portanto, medição do tamanho funcional é aplicável a um subconjunto de manutenções adaptativas. Isso inclui as funcionalidades do software adicionadas, alteradas ou excluídas bem como as funcionalidades do software fornecidas para converter dados e atender outros requisitos de conversão (ex.: relatórios de conversão).

É impraticável fornecer uma lista completa e abrangente de atividades de suporte e desenvolvimento. De certo modo, as seguintes áreas são identificadas como sugestões em relação à aplicabilidade da APF. É fortemente recomendado que cada organização desenvolva seu próprio guia com atividades, definições e terminologias específicas para a organização.

## Manutenção da Aplicação e Atividades de Suporte

Uma vez que as atividades de suporte e manutenção são assuntos para relatórios de inconsistência, diretrizes desenvolvidas localmente devem abordar estas áreas. A seguir estão algumas atividades mais comuns encontradas com sugestões de tratamento relativas à APF.

**Solicitações de Manutenção**

Independente da duração ou nível do esforço do trabalho solicitado, este é o tipo de atividade que determina como o trabalho é classificado. A APF não deve ser utilizada para medir trabalho de manutenção perfectiva ou corretiva. A manutenção corretiva deve ser contabilizada no projeto de desenvolvimento ou melhoria que introduziu o defeito. A manutenção perfectiva não deve ser contabilizada a nenhum projeto de desenvolvimento ou de melhoria.

Pode haver uma tendência em monitorar algumas funcionalidades de melhoria como trabalho de manutenção, mas este trabalho deve ser monitorado e reportado separadamente. A justificativa mais comum para inclusão é tanto para imediatismo ou para conveniência. Organizações freqüentemente fornecem uma forma mais rápida para pequenas solicitações de melhoria, normalmente 40 horas ou menos, a fim de reduzir o esforço de gerenciamento do projeto. Quando os requisitos de negócio são afetados, a APF deve ser aplicada para a medição dos resultados.

Se uma versão contém um misto de requisitos de manutenção adaptativa, corretiva e/ou perfectiva, deve ser tomado cuidado na separação do esforço de trabalho, uma vez que as últimas duas categorias não contribuem com Pontos de Função para o negócio. Embora a separação do esforço do trabalho possa ser relativamente fácil durante a fase de construção, dependendo do nível de detalhamento no controle do esforço, é normalmente mais difícil durante a maioria das fases finais de teste. Uma possível abordagem seria um rateio da versão inteira baseada no conteúdo proporcional.

Um projeto envolvendo apenas atualização de plataforma, linguagem, ou ambiente técnico, sem alteração nas funcionalidades do usuário, não deve ser submetido a uma medição do tamanho funcional da melhoria.

| Atividade | Dentro do escopo da contagem de melhoria |
|---|---|
| Correção de erros em Produção (Break/Fix) | Não |
| Manutenção perfectiva ou preventiva | Não |
| Atualização de plataforma, nova release do software do sistema | Não |
| Projeto tanto com melhorias e manutenções corretivas | Parcialmente |

## Solicitações Eventuais (Ad Hoc)

Funcionalidades que são fornecidas ao usuário final na forma de relatórios de execução única ou eventual e extração de dados, são certamente contáveis. A decisão para contar deve ser feita baseada em se as funções serão mantidas e a necessidade de negócio que o tamanho funcional vai atender. Deve ser observado que esta discussão é limitada a relatórios ou extrações produzidas pela Área de Desenvolvimento e não cobre a geração de relatórios eventuais ou extrações criadas pelo Usuário. Deve ser observado que a metodologia utilizada para produzir um relatório eventual não é normalmente tão rigorosa quanto um projeto de melhoria. Portanto, um cuidado deve ser tomado ao comparar o custo relativo de um trabalho com aquele da atividade geral de melhoria.

| Atividade | Dentro do escopo da contagem de melhoria |
|---|---|
| Relatórios executados uma vez | Convenção local |
| Atualização de tabelas | Não |
| Configuração de tarefa (Job) | Não |
| Correção de Dados | Não |
| Mudanças em massa de dados | Sim - como conversão se associado ao projeto. |

## Suporte ao Usuário Final

Qualquer esforço de trabalho extra-projeto relacionado a atividades classificadas como "não contáveis" devem ser contabilizados a uma classificação de trabalho que não seja de Novo Desenvolvimento ou de Melhoria. Para a Estimativa Preliminar ou Estudo de Viabilidade, o problema é que os requisitos do usuário ainda não estão bem definidos. Da mesma forma, um projeto neste estágio normalmente ainda não tem verba (e pode nunca ter). Na melhor das hipóteses, uma estimativa preliminar do tamanho em Pontos de Função pode ser determinada, mas nenhuma medida quantitativa deve ser aplicada neste ponto. Qualquer resultado obtido é apenas para propósito de orçamento e planejamento.

Geralmente atividades de suporte ao usuário extra-projeto, como uma resposta à pergunta "E se...?" e ajuda a usuários, não devem ser assuntos para APF.

| Atividade | Dentro do escopo da contagem de melhoria |
|---|---|
| Estimativa Preliminar e Estudo de Viabilidade | Na melhor das hipóteses, uma estimativa de ordem de grandeza |
| Respondendo "E se...?" | Não |
| Suporte Geral ao Cliente Extra-Projeto | Não |
| Suporte Help-desk | Parcialmente |

## Conclusão

Este capítulo pretende apenas para fornecer diretrizes e não é para ser interpretado como um conjunto de regras absolutas. Exceções a estes cenários podem sempre ser encontradas. Cada caso deve ser avaliado baseado nos requisitos do usuário e situação do negócio. Adicionalmente, cada organização de TI tem suas atividades exclusivas e seus processos de desenvolvimento. Estes podem ser abordados em um Guia de Pontos de Função local, com direcionamento de todos os aspectos da APF, incluindo premissas da contagem e qualquer interpretação diferenciada das regras do CPM.

# Resumo

Enquanto a maioria dos recursos de Desenvolvimento de Aplicações de TI estão habitualmente dedicados ao suporte ou melhoria das aplicações existentes, a maioria dos exemplos da Parte 3 estão direcionados a novos desenvolvimento. Este capítulo fornece diretrizes para o que contar e compreensão do processo de identificação em um ambiente de melhoria.

O primeiro passo é identificar corretamente as Funções de Dados e de Transação que devem ser incluídas, alteradas ou excluídas baseadas nos requisitos do usuário. Para modificações, a mais freqüente área observada são mudanças envolvendo apenas lógica de processamento interna. A Parte 1 inclui definições de lógica de processamento que são úteis na identificação de modificações nas funções de transação que devem ser contadas. Por último, cada Função de Dados e de Transação deve ser contada de acordo com as regras da Parte 1.

Também é importante separar as diferentes manutenções e atividades de suporte e seus esforços associados conforme discutido na Seção de Considerações sobre Manutenção e Melhoria.

# Parte 3 Capítulo 5

# Atividade de Conversão de Dados

**Introdução** Esta seção aborda a funcionalidade que será avaliada quando existem requisitos para migrar ou converter os dados em conjunto com um novo desenvolvimento ou projeto de melhoria ou para trocar uma aplicação para uma plataforma diferente. Parte 4 do CPM fornece outros exemplos de funções de dados e funções transacionais para conversão de dados.

**Conteúdo** Este capítulo irá discutir o seguinte como ilustrações de diferentes cenários de conversão:

# Funcionalidade de Conversão

Conversão de dados da aplicação é baseada na visão do usuário dos dados. Os usuários identificam os requisitos de dados com base em necessidades distintas, tais como Emprego, Contabilidade, Clientes ou Dados de Inventário. A visão do usuário destes dados abrange todos os atributos associados com o grupo de dados, tal como definido na aplicação. Este grupo de dados reconhecível pelo usuário e os dados associados atributos tornam-se a base para um grupo lógico de dados que cumpre uma exigência específica do usuário. Este é um arquivo lógico que exige que todos os seus atributos de dados devem ser mantidos como parte do todo (ligados e não independentes).

Atributos adicionais podem ser necessários por causa de exigências de negócios novas ou alteradas. Como parte da melhoria, pode ser necessário para converter e popular os atributos de dados adicionados como parte do projeto de melhoria. A visão do processo de conversão baseia-se na aplicação original, os arquivos lógicos que estão sendo convertidos e os requisitos de dados da nova aplicação.

O processo de conversão é executado contra todos os dados como visto pelo usuário para criar um arquivo lógico atualizado que cumpre os requisitos específicos do usuário para os novos / convertido dados da aplicação.

Aplicar as regras de identificação de PE padrão para identificar a funcionalidade de conversão. O processo elementar inclui todos os relatórios de exceção, os relatórios de erros, relatórios de conversão ou relatórios de controle necessários para garantir a integridade dos dados que estão sendo convertidos. Os ALIs da aplicação nova ou alterada, são populados com os dados convertidos e os requisitos de usuário determinam o que é exigido a partir da aplicação antiga para cumprir os requisitos funcionais do usuário do projeto.

## Cenário 1: Conversão de Dados em Projetos de Melhoria

O projeto envolve a integrar em uma aplicação corporativa a função Habilidade em RH que uma divisão da empresa já tinha implementado como uma aplicação stand-alone. Há um requisito para capturar uma única vez todos os dados existentes de habilidade e preencher os dados existentes atributos em um ALI em um aplicativo de RH existente. Os dados de habilidades existentes a serem importados serão contados como uma EE. Há um relatório de controle e um relatório de erro que são gerados para garantir a integridade da migração. Este processo será executado como parte da implantação da nova funcionalidade. Há um processo elementar para a carga inicial dos novos atributos de dados em um ALI do sistema de RH, incluindo os relatórios de controle e de erro. O processo de conversão será contado como uma EE, que será incluído no Tamanho Funcional do Projeto de Melhoria, mas não será adicionado ao Tamanho Funcional da Aplicação porque o processo é executado uma única vez.

## Cenário 2: Conversão de Dados com AIEs Referenciados

O usuário solicitou que um ALI (ou parte de um ALI) seja populado de um ALI de outra aplicação. Nesse exemplo, foi solicitado validar os dados com um outro ALI de uma terceira aplicação. Isso é especificado como um processo que será executado uma única vez e os dados referenciados na terceira aplicação não serão utilizados no futuro.

Os atributos a serem carregados servem como uma transação de entrada para popular o ALI que está recebendo e será contada como uma EE. Os dados referenciados na terceira aplicação para validação serão contados como um AIE e um ALR adicional. O ALI que está recebendo também será considerado como um ALR. Tanto a EE quanto o AIE devem ser incluídas na medição do projeto mas não devem ser adicionados ao tamanho funcional da aplicação.

## Cenário 3: Atribuição de Valores Padrão

Um projeto de melhoria solicita a inclusão de um DER em um ALI existente. O novo DER será populado com um valor padrão específico. Embora o ALI e qualquer transação modificada forem contadas como alteradas, não é contada uma funcionalidade de conversão. Nenhum dado atravessa a fronteira para estabelecer o valor padrão.

## O Que Não É Funcionalidade de Conversão

Esta seção descreve vários casos que não são considerados Conversões..

- Não conte atualizações de software devido à instalação de uma versão revista de pacotes de fornecedores como funcionalidade de conversão.
- Não conte a migração de uma aplicação para uma nova plataforma como uma funcionalidade de conversão.
- Não conte a conversão de dados realizada através de um utilitário de carga existente. Nenhuma funcionalidade foi desenvolvida para realizar a conversão.
- Mesmo que um AIE para a aplicação que está sendo medida é alterado, não pode haver qualquer funcionalidade de conversão. Apenas a aplicação que tem contada a função de dados como um ALI pode contar com a funcionalidade de conversão.

# Resumo

Quando um ALI é adicionado ou modificado, existe a possibilidade que um processo de conversão possa ser solicitado para popular o novo ALI ou DER(s) em um ALI existente. Parte da análise é identificar o que está atravessando a fronteira da aplicação. No caso de novos desenvolvimentos, o(s) depósito(s) de dado(s) existente(s) ou ALI(s) do(s) sistema(s) sendo substituídos é considerado como cruzando a fronteira da aplicação. Quando uma melhoria envolve alterações em ALI e uma lógica de processamento é solicitada para popular o novo atributo (ex.: validações, comparações lógicas, etc.), o ALI existente pode ser considerado como cruzando a fronteira da aplicação com uma EE. Se um novo atributo em um ALI é populado somente com um valor padrão ou nulo, a conversão não deve ser contata porque nada atravessa a fronteira da aplicação.

# Índice da Parte 3

Página intencionalmente deixada em branco