FGV EMAp

João Pedro Jerônimo

Modelagem Informacional

Revisão para A1

Rio de Janeiro

2025

Conteúdo

Modelagem Informacional de Requisitos (MIR)

Muitos (incluindo eu) entraram na matéria sem ter uma noção bem do que ela se tratava (Eu perdi as primeiras aulas, então ficou ainda pior). Mas saímos da matéria de Banco de Dados, ainda mexemos com eles, mas antes, estudávamos como eles eram armazenados, o que era necessário para se ter um local estruturado para seu armazenamento. Agora, vamos estudar como eles são usados, como podemos utilizar eles de forma eficiente a responder nossas dúvidas e ser utilizados em sistemas que iremos desenvolver.

Quando estamos montando um sistema, estamos interessados, principalmente em aplicações complexas, em saber exatamente o fluxo de informações. Quando o meu usuário fizer uma ação X no meu sistema, que informações eu preciso enviar para ele? Que informações ele vai me mandar? E para onde essas informações vão? Pensando nisso, foi criado o **Diagrama de Uso**

Definição 1.1 (Diagrama de Uso): Um diagrama de uso descreve as expectativas do público-alvo do meu projeto e clarifica o processo de identificação de requisitos. Pode responder as perguntas:

- O que está sendo descrito? Que sistema está sendo modelado?
- Quem interage com meu sistema?
- O que os **atores**(papéis) podem fazer?

Exemplo:

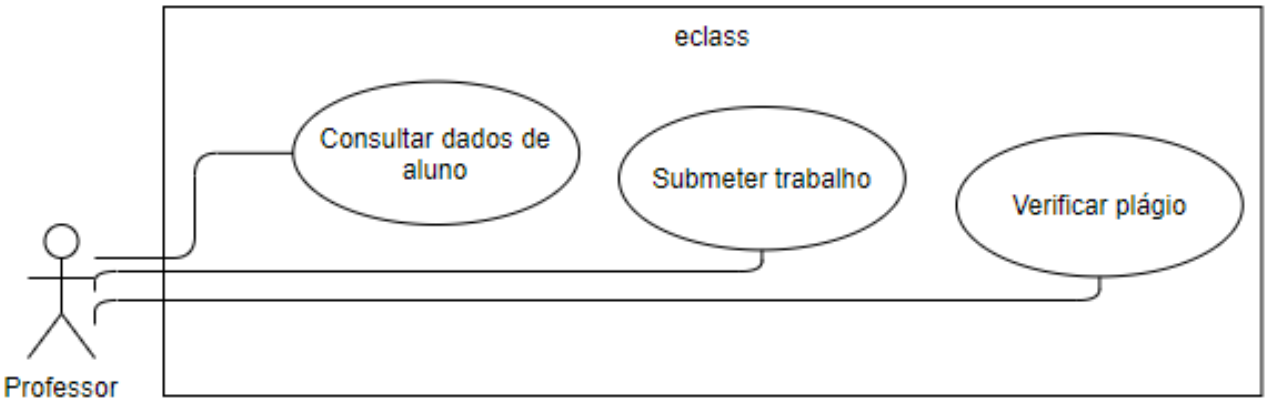


Figura 1: Exemplo de Diagrama de Uso

Definição 1.2 (Ator): Elementos fora do sistema que tem alguma importância no ecossistema do projeto

Definição 1.3 (Caso de Uso): Descrevem as funcionalidades esperadas de um sistema em desenvolvimento. Descrevem as expectativas da parte interessada no sistema.

Um ator interage com um caso de uso quando:
- Utilizam dos **casos**
- São utilizados pelos **casos**

Também podemos classificar os meus **atores**:
- **Humano**
- **Não-humano**
- **Primário**

    - Principal beneficiado da execução do **caso de uso**
- **Secundário**
    - Não recebe benefícios diretos
- **Ativo**
    - Inicia diretamente o **caso de uso**
- **Passivo**
    - Propicia funcionalidades para a execução do **caso de uso**

Podemos também classificar as formas de como os atores se relacionam com os casos de uso

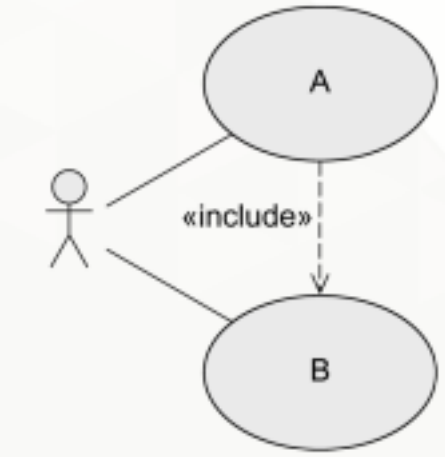


Figura 2: Necessita que **(B)** seja executado antes de **(A)**

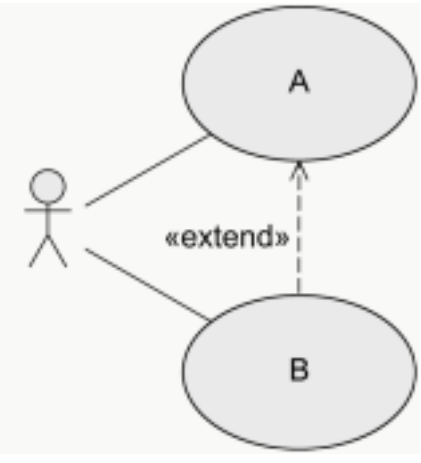


Figura 3: **(A)** pode executar sozinho e decide se **(B)** vai executar ou não, sendo **(B)** uma extensão (das funcionalidades) do caso **(A)**

Assim, conseguimos montar uma pequena estruturação para os Casos de Uso

- **Nome**
- **Descrição**
- **Pre-condições**: Pré-requisitos para que o caso de uso funcione corretamenteo
- **Pós-condições**: Estado esperado do sistema após a execução do caso de uso
- **Situações de erro**: Erros relevantes
- **Estado do sistema na decorrência de um erro**
- **Atores que comunicam com o caso de uso**
- **Gatilho de acionamento**: Eventos que executam o caso de uso
- **Processo Principal**
- **Processos Alternativos**: Possiveis desvios do processo principal

Definição 1.4 (Informação): Dado interpretado segundo um contexto

Definição 1.5 (Processo): Conjunto de atividades logicamente organizadas e condicionais, cuja execução visa alcançar um objetivo determinado

Definição 1.6 (Comportamento): Trajetória percorrida por um processo. Todo efeito observável no ambiente externo do processo

Porém esse esquema de modelagem tem alguns pontos negativos e limitações:
- Ênfase excessiva no detalhamento do comportamento sistêmico

- Falta de regra objetiva para orientar os níveis de abstração
- Insuficiente detalhamento da informação que flui entre o sistema e o ambiente

Surge então o MIR para que possa consertar esses problemas, surgindo como uma especialização do Diagrama de Uso. Essa solução leva alguns princípios em consideração
- Focar nos objetivos
- Atribuir níveis de abstração aos objetivos
- Focar no detalhamento da informação

> **Definição 1.7** (Objetivo Informacional): Objetivos dos atores que geram eventos externos que exigem **intervenção atômica** do sistema com trocas de informação capazes de mudar o estado do ambiente, do sistema ou de ambos

> **Definição 1.8** (Intervenção Atômica): Se processa sem interrupções ou temporizadores. Uma vez concluída, coloca o sistema em estado de espera para o próximo evento

Então montamos uma tabelinha de forma que cada coluna representa um ator e cada linha um objetivo informacional associado ao ator

> *Exemplo (MOBI Taxi)*: A cooperativa MOBITAXI deseja construir um aplicativo de celular para atender seus passageiros. Todos os motoristas e clientes precisam estar cadastrados com nome, número de telefone e endereço. O passageiro pode chamar o táxi de qualquer local dentro do estado do Rio de Janeiro. A posição (GPS) do celular do cliente indicará o local aonde o motorista deverá buscá-lo. Para evitar concorrência predatória entre os colegas taxistas e diminuir o tempo de espera do cliente, o sistema deverá escolher e direcionar a corrida ao motorista mais próximo. Ao final da corrida, o valor será debitado do cartão de crédito do cliente e creditado na conta do motorista. Sabe-se de 1% de todas as corridas são da administração. O cliente ainda poderá avaliar a corrida pontuando-a entre 0-5, além de escrever um comentário. A administração da cooperativa poderá "solicitar" a saída de motoristas mal avaliados
>
> Estes são os requisitos do meu sistema (Banco de Dados):
>
> 1.Manter cadastro de motoristas e colaboradores (nome, celular e endereço), um motorista pode possuir mais de um veículo, assim como pode emprestá-lo;
>
> 2.Manter cadastro de veículos (marca, modelo, ano, placa), um veículo pode ser conduzido por mais de um motorista;
>
> 3.Manter cadastro de clientes (nome, celular e endereço), um cliente pode possuir mais de um endereço;
>
> 4.Manter cadastro de viagens feitas pelo cliente, que só pode avaliar um motorista por corrida;
>
> 5.Manter registro de créditos da cooperativa, considerando que 1% dos créditos sustentam a administração;
>
> 6.A administração da MOBITAXI pode fazer uma avaliação dos motoristas a partir da #viagens realizadas e da pontuação dada pelos clientes. Além de poder acessar os contadores básicos: #motoristas, #clientes, e #viagens por mês
>
> Agora que temos toda a contextualização, podemos fazer a minha tabela

Motorista	Passageiro	Administração
(1) Ver oferta de corrida	(6) Pedir corrida	(10) Cadastrar Motorista
(2) Atender corrida	(7) Cancelar corrida	(11) Cadastrar Admin
(3) Recusar corrida	(8) Avaliar corrida	(12) Cadastrar Veículo
(4) Fechar corrida	(9) Atualizar cadastro	(13) Avaliar Motorista
(5) Atualizar cadastro		(14) Administrar Caixa

Com todos os meus objetivos informacionais bem-definidos, eu vou agora especificá-los ainda mais através da criação de uma interface informacional para **cada um deles**

Definição 1.9 (Interface Informacional): Define os fluxos de informação que entram e saem durante o processamento do objetivo pelo sistema e divide em duas etapas:

1. Especificação dos fluxos
   - Explicito detalhadamente as informações que fluem no objetivo, seja elas sendo recebidas por ele ou sendo enviadas para outro objetivo
2. Dicionários de itens elementares
   - Detalha as propriedades das informações que estão circulando no objetivo especificado

Exemplo (MOBI Taxi): Ainda utilizando da tabela que montamos anteriormente, vamos fazer as interfaces informacionais dos objetivos (1) e (2).

Ator: Motorista | Objetivo 1: Ver oferta de Corrida

← `oferta_corrida = id_cliente + nome_cliente + avaliação_cliente + ponto_partida_gps + dt_hr_pedido`

- Descrição: Informações de um passageiro pedindo uma corrida.
- Propósito: Dar ao motorista a opção de aceitar ou não a corrida.
- Frequência: 100/dia

Ator: Motorista | Objetivo 2: Atender Corrida

→ `aceite_corrida = id_motorista + nome_motorista + localização_gps + dt_hr_aceite`

- Descrição: Uma vez que o motorista aceitou a corrida, o tempo começa a contar até chegar ao passageiro e nenhum outro motorista pode pegar a mesma corrida.
- Propósito: Informar ao sistema a posição, as informações do motorista e a previsão de chegada ao passageiro.
- Frequência: 100/dia

A seta → apontando para dentro do texto significa que a informação está sendo **enviada para o sistema**, enquanto a seta ← apontando para fora do texto quer dizer que a informação está sendo **recebida** pelo ator. Com as interfaces criadas, podemos agora fazer os dicionários elementares de cada interface. Farei apenas da interface informacional (1)

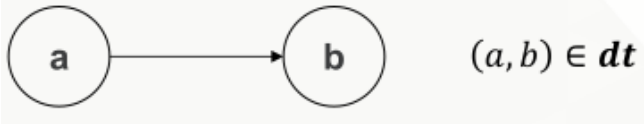
Ator: Motorista | Objetivo 1: Ver Oferta de Corrida			
Nome	**Descrição**	**Tipo**	**Domínio**

`Id_cliente`	Id do cliente	`Num. Sequencial natural`	Gerado automaticamente
`Nome_cliente`	Nome do cliente	`Str`	
`Avaliação_cliente`	Média de avaliação do cliente por outros motoristas	`Num. Natural`	$\{1..5\}$
`Ponto_partida_gps`	Posição geográfica (Lat/Long) do passageiro pedindo corrida	`Float`	Coordenadas WGS 84
`Dt_hr_pedido`	Data e hora em que o passageiro fez o pedido	`Timestamp`	

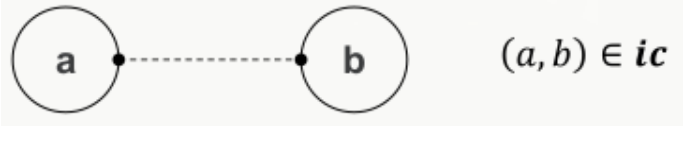
E para finalizar, temos os objetivos organizacionais

Definição 1.10 (Objetivo Organizacional): Uma sequência admissível de objetivos informacionais, representando uma linha de trabalho/objetivo de negócio relevante no domínio da aplicação para um ou mais atores

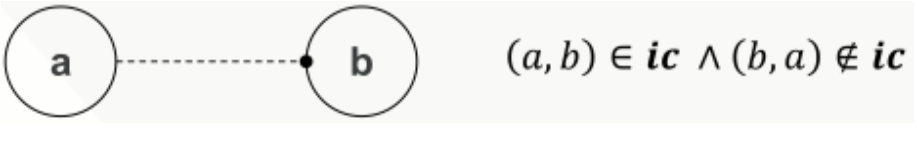
Definição 1.11: Sequências admissíveis de objetivos informacionais:



Sequência de Objetivos Informacionais em relação de dependência temporal (**dt**). Isso significa que o objetivo **b** só poderá ser executando quando **a** terminar de executar



Sequencia de objetivos informacionais em relação de incompatibilidade (**ic**). Isso significa que o evento **a** não ocorre se **b** ocorrer e vice-versa



Sequencia de objetivos informacionais em relação de incompatibilidade **ic** em apenas um sentido. Isso significa que, se **a** acontecer, **b** não pode ocorrer, porém o contrário não vale

Exemplo (MOBI Taxi): **Atender Passageiro**

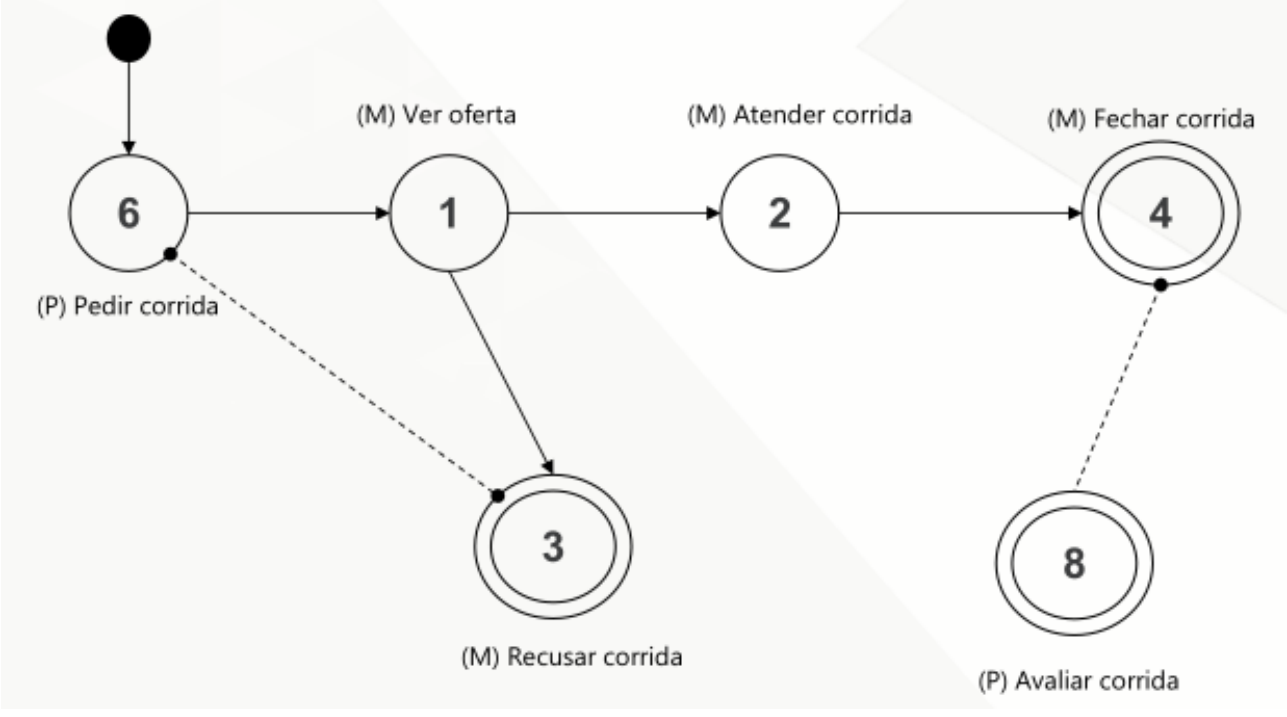


Figura 4: Exemplo de Objetivo Organizacional

- (6) $\rightarrow$ (1): O motorista só pode ver a oferta após o passageiro pedir a corrida
- (6) o—o (1): Não é possível recusar uma corrida sem que ela seja feita e um motorista não consegue recusar a corrida diretamente sem sequer a vê-la
- (8) —o (4): Não posso avaliar uma corrida que ainda não foi concluída, porém, mas o motorista pode fechar corridas independentemente do passageiro avaliá-la ou não

Data Warehouses

2.1 O que é?

Antes de adentrar no conceito de um Data Warehouse, vamos ter algumas definições antes

Definição 2.1.1 (Dado Operacional): Informações com pouco tempo de vida, utilizados essencialmente para o dia a dia e funcionamento do sistema, tem alta frequência e vão se atualizando conforme o tempo passa. Usado por diversos funcionários em setores diferentes e é orientado à aplicação

Definição 2.1.2 (Dado Analítico): Informações duradouras, que não se atualizam frequentemente e nem tem alta frequência de acesso. Redundância não é um problema, utilizado por um grupo ninchado de pessoas. Orientado ao assunto/contexto

A definição pode não ser muito clara, mas podemos imaginar um comparativo. Pense nos dados operacionais como um registro de diário, enquanto os dados analíticos são um livro de história. Os dados operacionais são frequentes, e não representam uma importância a longo prazo, por exemplo, o preço de um produto que está sendo passado no caixa, enquanto os dados analíticos representam um marco da empresa, como por exemplo, o lucro total da empresa na semana x.

Vamos também definir melhor:

Definição 2.1.3 (Orientado à Aplicação/Application Oriented): Dados projetados para suportar processos do dia a dia da organização. Estrutura de dados otimizada para inserções, atualizações e consultas rápidas de informações atuais com foco em transações (OLTP – Online Transaction Processing).

Exemplo: Sistema bancário para processar depósitos/saques, sistema hospitalar para registrar consultas.

Definição 2.1.4 (Orientado ao Assunto/Subject Oriented): Dados projetados para analisar informações de um assunto de negócio específico (clientes, vendas, faturamento, estoque, etc.). Dados integrados e organizados de forma a responder perguntas estratégicas. Foco em análise histórica e tendências (OLAP – Online Analytical Processing). Normalmente são read-only (só leitura, não ficam sendo atualizados a todo instante).

Exemplo: DW que reúne anos de dados de vendas para gerar relatórios, dashboards ou prever demanda.

Imagine que você está fazendo o sistema, você sabe que seus dados precisam ser bem estruturados, o foco do seu sistema é ser duradouro e bem manutenível além de promover uma segurança boa, então seus dados vão ser **application-oriented**, de forma que você faz questão de deixar tudo muito bem-estruturado e separado. Porém imagine que o foco do seu sistema é responder perguntas de negócio de forma rápida e eficiente, por exemplo: “Quanto minha empresa faturou nos últimos 3 meses?”, dificilmente você vai se importar que as informações do banco que você está consultando estejam todas na norma 3, ou separadas direitinho com as chaves-estrangeiras bem definidas e organizadas, o importante é você responder a pergunta de forma rápida e eficiente, então você vai estruturar esse banco de forma a atingir esse objetivo, então seus dados são **subject-oriented**

Com isso em mente, agora podemos entender o que são Data Warehouses

Definição 2.1.5 (Data Warehouse): Um Data Warehouse é um **repositório estruturado** de dados **integrados**, **orientados ao assunto**, **com informações sobre toda a empresa**, **históricos** e **variantes com o tempo**. O propósito de um Data Warehouse é a extração de **dados analíticos**. Pode armazenar dados detalhados e/ou resumidos

De forma resumida, um Data Warehouse também é um banco de dados, porém, estruturado e com um objetivo totalmente diferente dos bancos convencionais que estudamos na disciplina de **banco de dados**. Mas o que compõe um Data Warehouse? No núcleo, ele em si é apenas esse banco de dados diferenciado, porém, como funciona o sistema? Como é estruturado um projeto/sistema em que um Data Warehouse é incluído?

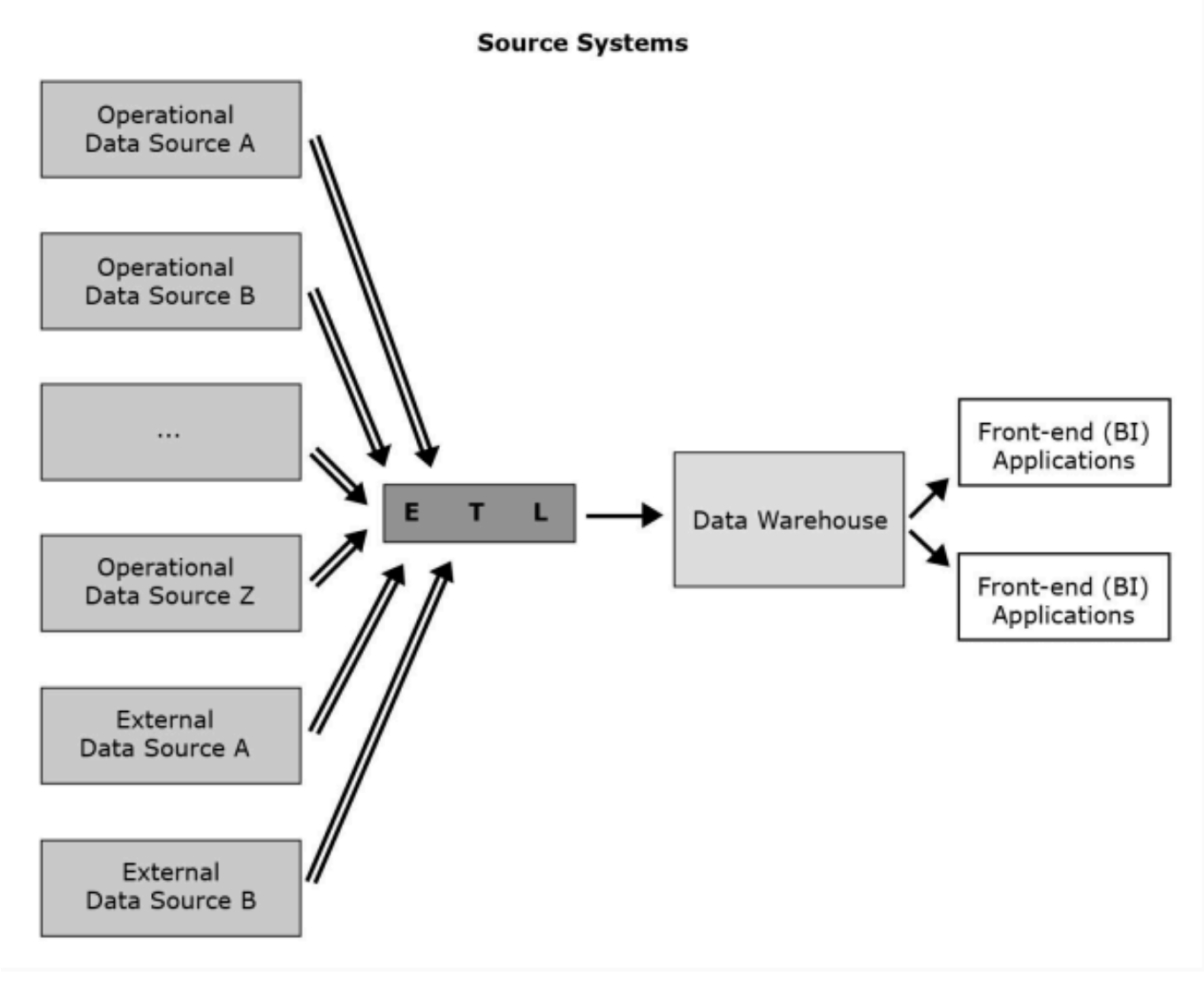


Figura 5: Ecossistema Data Warehouse

Todas as definições abaixo são referentes a elementos apresentados na imagem como componentes de um ecossistema de Data Warehouse

Definição 2.1.6 (Sistemas de Origem): Sob o contexto de DW, os sistemas de origem são bases de dados operacionais e outros repositórios de dados (qualquer conjunto de dados usado para proposta operacional) que provê informação analítica útil para assuntos de análise. Cada unidade de armazenamento de dados operacionais que é usada como sistema de origem tem duas finalidades:

- A própria finalidade operacional
- Ser a fonte do DW

Sistemas de origem podem incluir fontes internas e externas

Definição 2.1.7 (Infraestrutura de ETL): Facilita a leitura dos dados dos sistemas de origem para o Data Warehouse. O ETL (Extract, Transform, Load) tem as seguintes tarefas:

- Extrair dados analiticamente úteis das origens operacionais
- Transformar tais dados de forma aderente a estrutura de "orientação-ao-assunto" do modelo do DW (ao mesmo tempo que assegura a qualidade dessa transformação)
- Carregar os dados transformados e com qualidade assegurada para o destino "target" data warehouse

Definição 2.1.8 (Data Warehouse): As vezes referenciado como "target system". Um DW típico lê as informações analiticamente úteis dos sistemas de origem de forma periódica ou contínua ,provendo sempre dados atualizados para análise

Definição 2.1.9 (Aplicações Front-end): De forma similar aos bancos operacionais, as aplicações "front-end" permitem o acesso indireto dos usuários aos dados

Como vimos, um data warehouse tem informações sobre **toda** uma empresa, podendo gerar análises sobre todos os seus departamentos. Mas e se isso for um comportamento indesejado? Eu quero ter uma análise detalhada, mas focar apenas no meu departamento e evitar que os outros vejam informações que não deveriam ou informações que possam atrapalhar nas suas análises, então aí entram os Data Marts

Definição 2.1.10 (Data Mart): Os Data Marts Seguem os mesmos princípios de um DW ,mas possuem um escopo mais limitado, geralmente ligado a um uso mais departamental, e não de forma abrangente à empresa conforme um DW.

- **Data Mart independente**: Lobo solitário, criado como se fosse um DW. Um DM independente tem os próprios sistemas de origem e infraestrutura de ETL
- **Data Mart dependente**: Não possui os próprios sistemas de origem. Os dados são lidos de um DW

2.2 Como um Data Warehouse é estruturado?

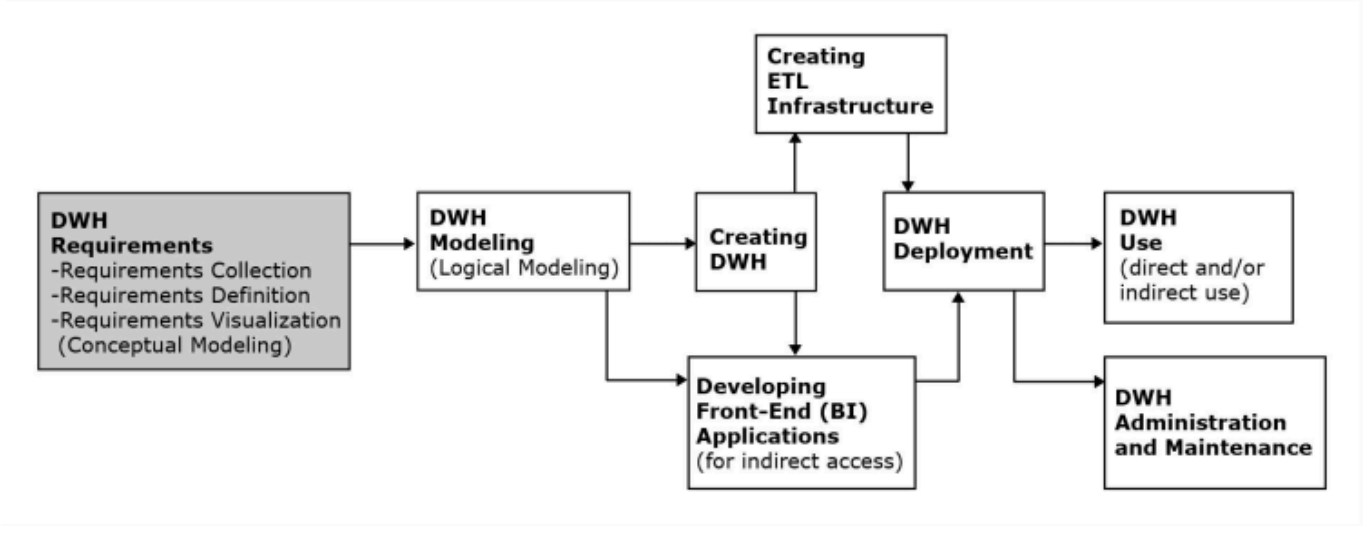


Figura 6: Passo-a-passo da montagem de um Data Warehouse

2.2.1 DWH Requirements

Elicitação de requisitos, definições e visualização – deve especificar as capacidades e funcionalidades desejadas do futuro DW

- Os requisitos são embasados nas necessidades analíticas que podem ser supridas pelos dados dos sistemas de origem disponíveis, tanto internos quanto externos
- A coleta de requisitos é feita através de entrevistas com vários stakeholders
- Além das entrevistas, métodos adicionais para elicitação de requisitos podem ser usados, tais como grupos focais, surveys, observações de processos existentes, etc
- A coleta de requisitos deve ser definida e escrita claramente em um documento, e então visualizada como um modelo de dados conceitual.

2.2.2 DWH Modeling

Modelagem do data warehouse (modelagem lógica do DW) – criação do modelo de dados do DW que é implementável em um SGBD

- Logo após a etapa de coleta de requisitos inicia-se a modelagem do DW, de onde serão criados modelos implementáveis no SGBD escolhido.
- O Modelo de dados lógico ou modelo de dados de implementação leva em conta a lógica particular do software de gerenciamento de banco de dados escolhido.

2.2.3 Creating DWH

Criação do DW – usar um SGBD para implementar o modelo de dados do DW

- Tipicamente, os data warehouses são implementados usando os mesmos SGBDs dos bancos operacionais, tais como MS SQL, Oracle,
- No entanto, existem SGBDs especializados em processar grandes volumes de dados, tais como Teradara e Vertica

2.2.4 ETL Infraestructure

- Criar procedimentos e códigos para:
  - Extração automática de dados relevantes das fontes de dados operacionais
  - Transformação dos dados extraídos, de forma que a qualidade seja assegurada e a estrutura seja aderente ao modelo de DW implementado
  - A carga contínua dos dados transformados para dentro do data warehouse

- Devido à quantidade de detalhes que devem ser considerados, a criação da infraestrutura ETL costuma ser a parte que mais consome tempo e recursos do processo de desenvolvimento do data warehouse.

2.2.5 Developing Front-End Applications

Desenvolvimento de aplicações front-end (BI) - projetar e criar aplicativos para uso indireto pelos usuários finais

- Os aplicativos front-end estão incluídos na maioria dos DW e são frequentemente chamados de aplicativos de inteligência de negócios (BI)
- Os aplicativos front-end contêm interfaces (como formulários e relatórios) acessíveis por meio de um mecanismo de navegação (como um menu)

2.2.6 DWH Deployment

Liberar o DW e seus aplicativos front-end (BI) para uso dos usuários finais

2.2.7 DWH Use

Uso do data warehouse - a leitura dos dados no data warehouse

- Uso indireto
  - Via front-end (BI) applications
- Uso direto
  - Via SGBD
  - Via OLAP (BI) tool

2.2.8 SWH Administration and Maintenance

Administração e manutenção do data warehouse - realizar atividades que dão suporte ao usuário final do DW, incluindo o tratamento de questões técnicas, tais como:

- Fornecer segurança para as informações contidas no data warehouse
- Garantir espaço suficiente no disco rígido para o conteúdo do data warehouse
- Implementar os procedimentos de backup e recuperação

2.3 Modelo Dimensional

Vimos na disciplina de banco de dados sobre **modelos relacionais**, porém, estudiosos, posteriormente, perceberam que esse modelo é ineficiente para **análise de dados**, então foram criados modelos diferentes para a criação desses bancos. Então como podemos fazer?

- **Kimbal**: Star Schema (Mais utilizado)
- **Inmon**: Floco de neve

O método do **Kimbal** é o mais utilizado na prática, pois ele é uma versão não-normalizada do **Inmon**, já que, em um Data Warehouse, a normalização aparenta ser desnecessária, já que todas as entradas são apenas de leitura

Antes nós montávamos como **entidade** e **relacionamentos**, agora, nossos elementos são:

- **Dimensões**
- **Fatos**

> *Exemplo*: Dentro da nossa loja, queremos criar um dashboard para gerenciar e entender as devoluções dos produtos feitos. De um modo bem simples, vamos ter o fato **Devolução**, ela ocorreu e pronto, mas o que ela engloba que não está necessariamente dentro do fato da devolução? Temos o produto, temos o calendário (Quando a devolução foi feita) e tem o motivo da devolução (Tem mais coisas, mas vamos simplificar por aqui). A primeira vista, essas coisas estão dentro da devolução, correto? Mas se pararmos para pensar, vários pro-

dutos podem ser devolvidos de maneiras independentes, assim como nem toda devolução tem um motivo diferente

Definição 2.3.1 (Tabela de Dimensões): Contém descrições de negócios, organização, ou empresas no qual o sujeito da análise pertence. As colunas na tabela dimensional costumam ter informações descritivas, como textos (e.g, `product_color`, `product_description`, `client_name`). Essas informações providenciam uma base para a análise do sujeito

Definição 2.3.2 (Tabelas de Fatos): Contém medidas relacionadas ao sujeito da análise e chaves-estrangeiras que ligam os fatos às tabelas de dimensões. As medidas na tabela de fatos costumam ser numéricas com intenção de análises computacionais e matemáticas

2.4 Star Schema

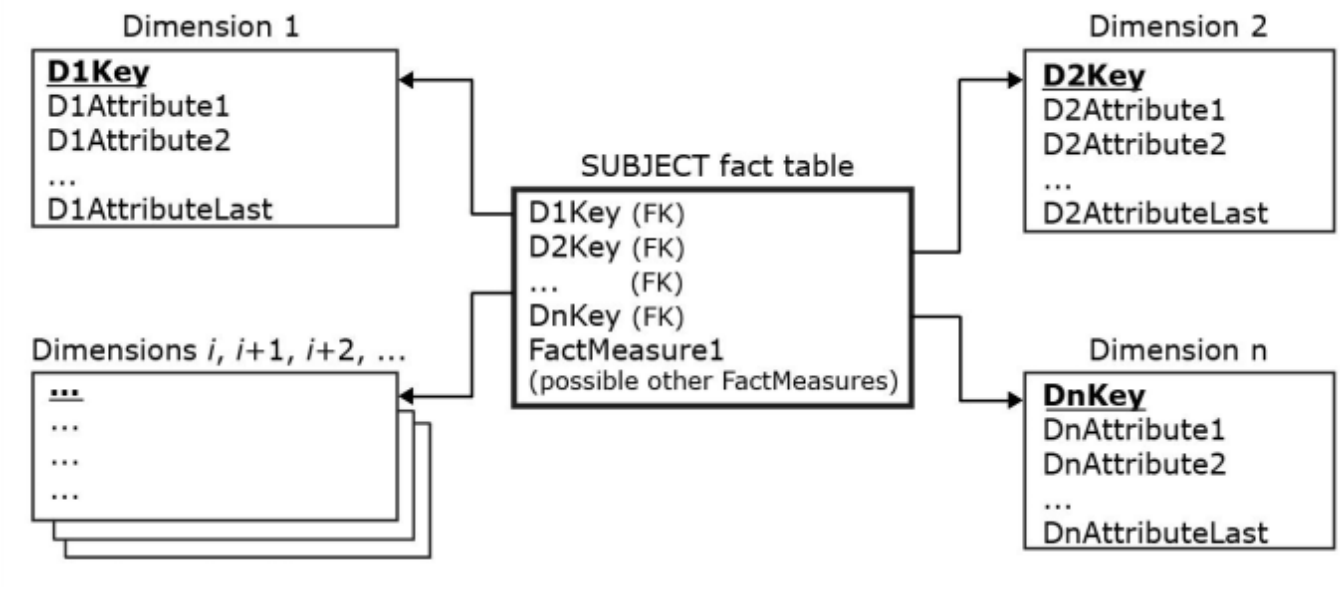


Figura 7: Exemplo de estruturação em **Esquema Estrela**

Os **Fact Measures** são informações dentro do fato que **não se aplicam em nenhuma dimensão**

Exemplo: Queremos criar o fato **venda**, ele engloba as dimensões de **produto**, **cliente**, **loja**, **calendário** e **vendedor**. Porém, não faz sentido, por exemplo, colocar a quantidade de produtos comprados em **nenhuma dimensão**, ou o **valor total da compra**, de forma que essas informações são localizadas única e exclusivamente no fato

Vale ressaltar que, dado a dimensão i, a sua chave-primária **não é a mesma chave-primária do modelo relacional**, pois a repetição dos registros pode ocorrer sem problema nenhum. Como aplicar chaves-primária nas dimensões e nos fatos será visto posteriormente

2.5 Como um fato se organiza?

Uma tabela de fatos tem:

- Chaves-estrangeiras conectando a tabela de fato para as tabelas de dimensões
- As medidas relacionadas ao sujeito da análise

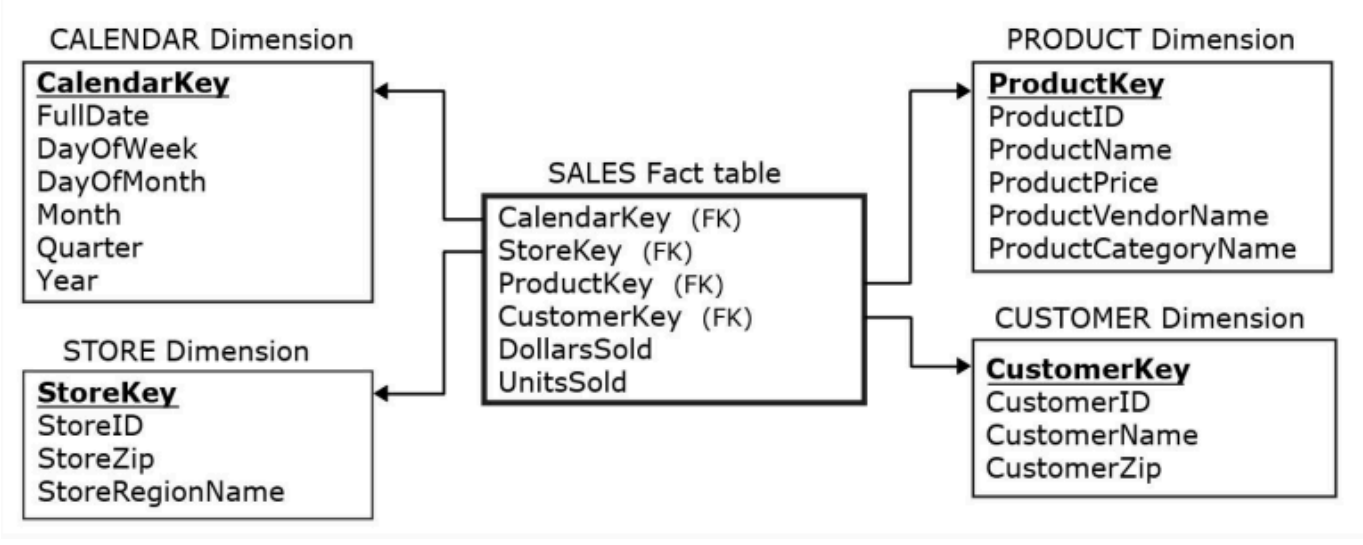


Figura 8: Exemplo do Esquema Estrela nas lojas zagi

Os principais pontos a se destacar é que, no modelo estrela, nós não colocamos o **id** do modelo relacional como a chave-primária da dimensão. Por quê? Por conta que **não há normalização**, por conta disso, podem aparecer chaves-primárias repetidas, algo que **não pode acontecer**. Então criamos as **chaves substitutas** (Surrogate keys), que são chaves que não se repetem na tabela de dimensão (Não são as mesmas chaves do modelo transacional). Porém, o mesmo não acontece na tabela de fatos, ela não possui uma surrogate key, então o que diferencia um fato dos demais?

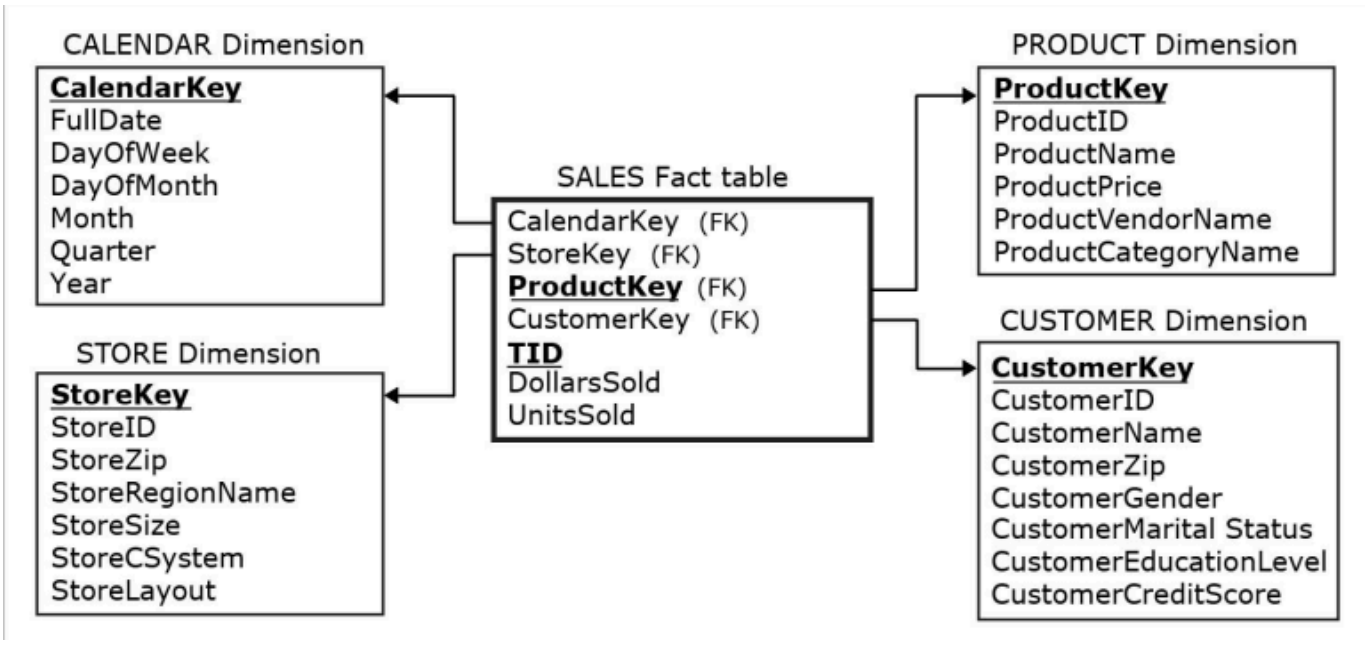


Figura 9: Exemplo correto do modelo dimensional das lojas zagi

Podemos ter algumas abordagens **arbitrárias** para identificar os fatos. Por exemplo, na imagem acima, nós diferenciamos dois fatos pelo **id da transação** e pela **surrogate key** do produto, já que um produto comprado só pode estar associado a **uma única transação**. Eu também poderia identificar um fato utilizando uma **chave composta** das **chaves-estrangeiras** das dimensões (Tem alguns problemas e questionamentos para esse exemplo em específico, mas vamos supor que não precisa de alterações a mais)

2.6 Galáxia de Estrelas

É um modelo que contém **vários fatos** que compartilham **dimensões** entre si

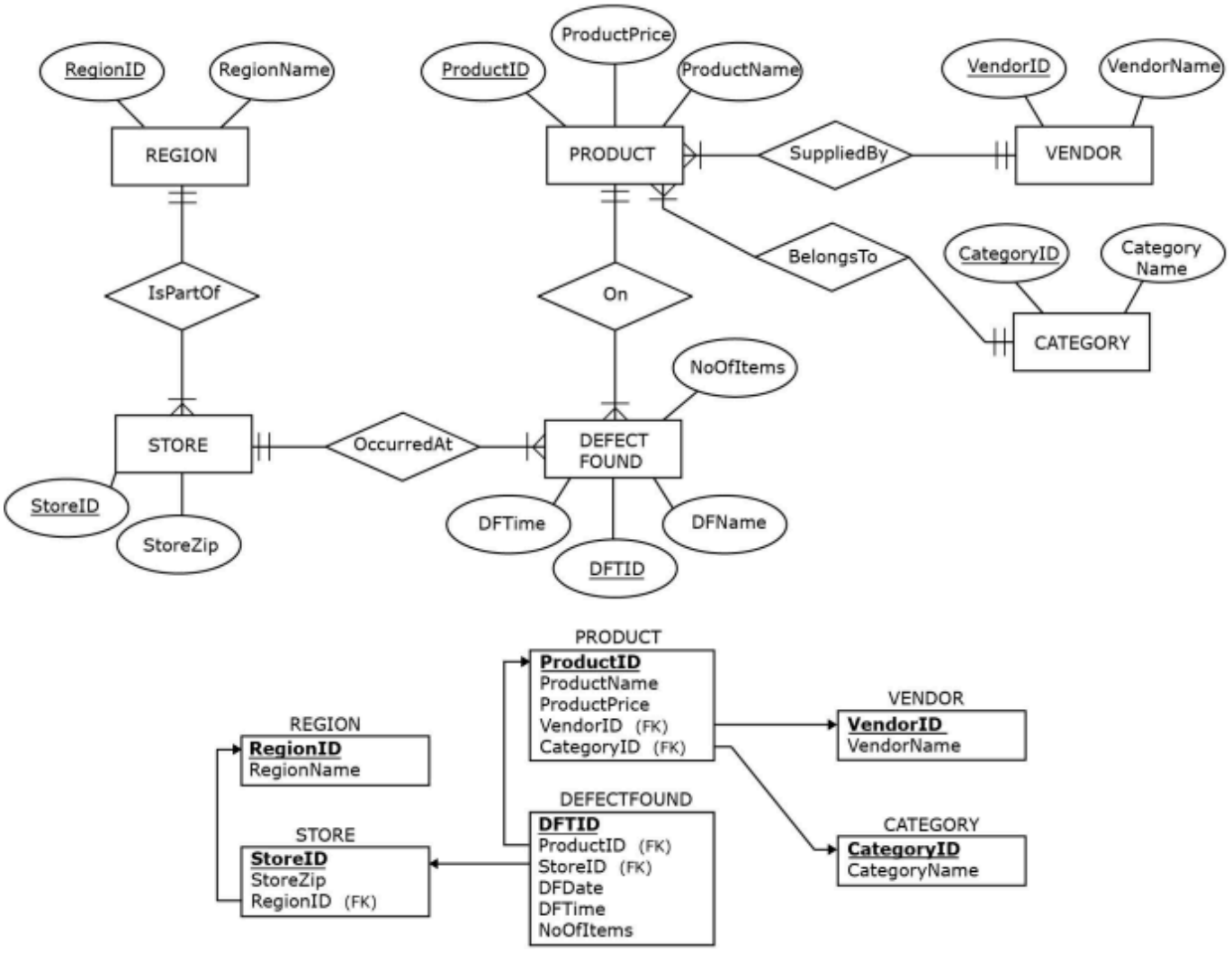


Figura 10: Modelo ZAGI adaptado

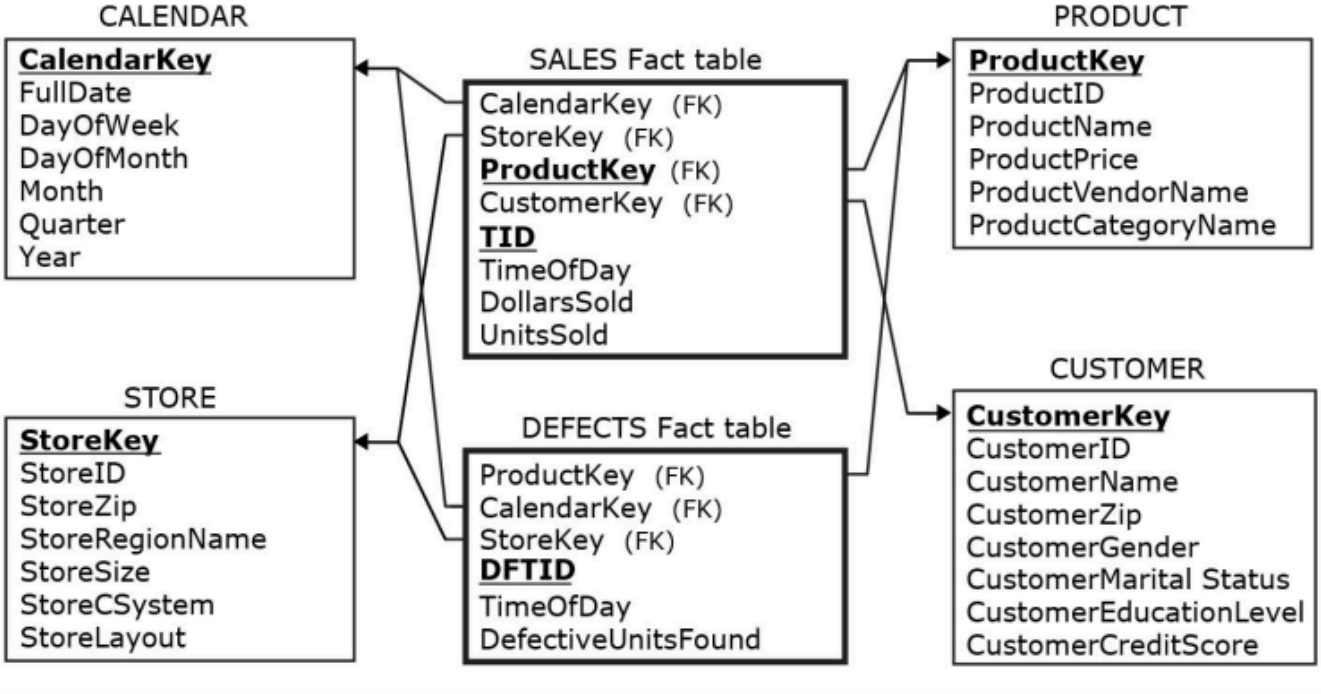


Figura 11: Galáxia ZAGI

2.7 Detalhamento de Fatos

Antes disso, precisamos entender melhor sobre a **granularidade** da tabela. A granularidade e refere a o que uma única linha de uma tabela se refere. Tabelas de fato com um refinamento alto de granularidade expressam um único fato, enquanto tabelas com granularidade maior expressam, em suma, um conjunto de fatos agrupado. Por conta desse detalhamento v.s agrupamento, ambas possuem suas vantagens e desvantagens

Com base na granularidade da tabela, podemos classificá-la em duas:

- **Line-item detailed fact table**: Cada linha representa uma linha de item de uma transação em particular
- **Transaction-level detailed fact table**: Cada linha representa uma transação em particular

Um exemplo fica melhor de compreender

> *Exemplo*: Vamos imaginar que temos um negócio de aluguel de carro, e que nós alugamos o carro de forma que o nosso cliente pode escolher alguns acessórios a mais (Por exemplo, ele vai pagar $R\$40,00$ adicionais para colocar uma cadeirinha de bebê). Como cada transação pode ter características diferentes, faz sentido que cada fato de transação tenha uma especificação dizendo os itens específicos que foram juntos, então teríamos uma **line-item detailed fact table**.
>
> Mas vamos supor que você não da essa opção aos clientes, e eles tenham que escolher baseado em categorias (Por exemplo, a categoria A é mais barata, não tem ar-condicionado, não tem gps, entre outras coisas e eu tenho outras categorias também), então não faz muito sentido adicionar **na tabela fatos** todas as coisas inclusas nas categorias, e sim adicionar isso na **dimensão** do fato, já que não é algo que varia de fato para fato. Nesse caso, teríamos uma **transaction-level detailed fact table**

2.8 Dimensões Lentamente Alteráveis

Normalmente as dimensões não conseguem mudar, porém, alguns casos elas podem. Por exemplo, o preço de um produto pode variar. O que fazemos nesses casos? Essas dimensões são chamadas de **dimensões lentamente alteráveis**. Então vamos temos 3 abordagens diferentes:

2.8.1 Tipo 1

Eu vou mudar o valor na própria tabela de dimensão, ou seja, o novo valor substitui o antigo. Não preserva uma história dessa dimensão

CUSTOMER

CustomerKey	CustomerID	CustomerName	TaxBracket
1	111	Linda	Low
2	222	Susan	Medium
3	333	William	High

CUSTOMER

CustomerKey	CustomerID	CustomerName	TaxBracket
1	111	Linda	Low
2	222	Susan	High
3	333	William	High

Figura 12: Exemplo onde o Imposto de Susan é alterado para **Alto**

2.8.2 Tipo 2

Cria uma nova entrada na dimensão usando uma nova **surrogate key** toda vez que o valor muda. Usado em casos que eu quero preservar a história do meu sistema. Porém eu preciso de um método para indicar qual entrada está **vigente**, então utilizamos atributos como **timestamps**

CUSTOMER

CustomerKey	CustomerID	CustomerName	TaxBracket
1	111	Linda	Low
2	222	Susan	Medium
3	333	William	High

CUSTOMER

CustomerKey	CustomerID	CustomerName	TaxBracket	Effective StartDate	Effective EndDate	Row Indicator
1	111	Linda	Low	1.1.2010	n/a	Current
2	222	Susan	Medium	1.1.2010	12.31.2019	Not current
3	333	William	High	1.1.2010	n/a	Current
4	222	Susan	High	1.1.2020	n/a	Current

Figura 13: Exemplo criando uma nova linha de Susan. Dependendo da situação que você se encontra, pode ser plausível **não utilizar** colunas de identificação temporal

2.8.3 Tipo 3

Agora adicionamos uma nova coluna de “anterior” e de “atual” para cada coluna que pode ser alterada. Aplicável quando há uma quantidade limitada de alterações possíveis em uma coluna (Por exemplo, se eu tenho uma única coluna de “anterior” e uma única coluna de “novo”, então sempre que eu atualizo a minha dimensão, eu eu vou perder a informação anterior à anterior da que eu atualizei agora, ou seja, se eu tinha que `anterior=1` e `atual=2` e eu atualizo novamente, então vou obter `anterior=2` e `atual=3`, de forma que eu perco o 1)

CUSTOMER

CustomerKey	CustomerID	CustomerName	TaxBracket
1	111	Linda	Low
2	222	Susan	Medium
3	333	William	High

CUSTOMER

CustomerKey	CustomerID	CustomerName	PreviousTax Bracket	Previous TaxBracket EffectiveDate	Current TaxBracket	Current TaxBracket EffectiveDate
1	111	Linda	n/a	n/a	Low	1.1.2010
2	222	Susan	Medium	1.1.2010	High	1.1.2020
3	333	William	n/a	n/a	High	1.1.2010

Figura 14: Exemplo registrando apenas duas mudanças. Dependendo da situação, você também pode **não fazer** colunas de **identificação temporal**

2.9 Modelo Floco de Neve (Snowflake)

Como dito anteriormente, a grande diferença dele pro anterior (Modelo estrela) é a presença de **normalização**, já que é muito defendido que normalização não é algo muito necesário para análise de dados

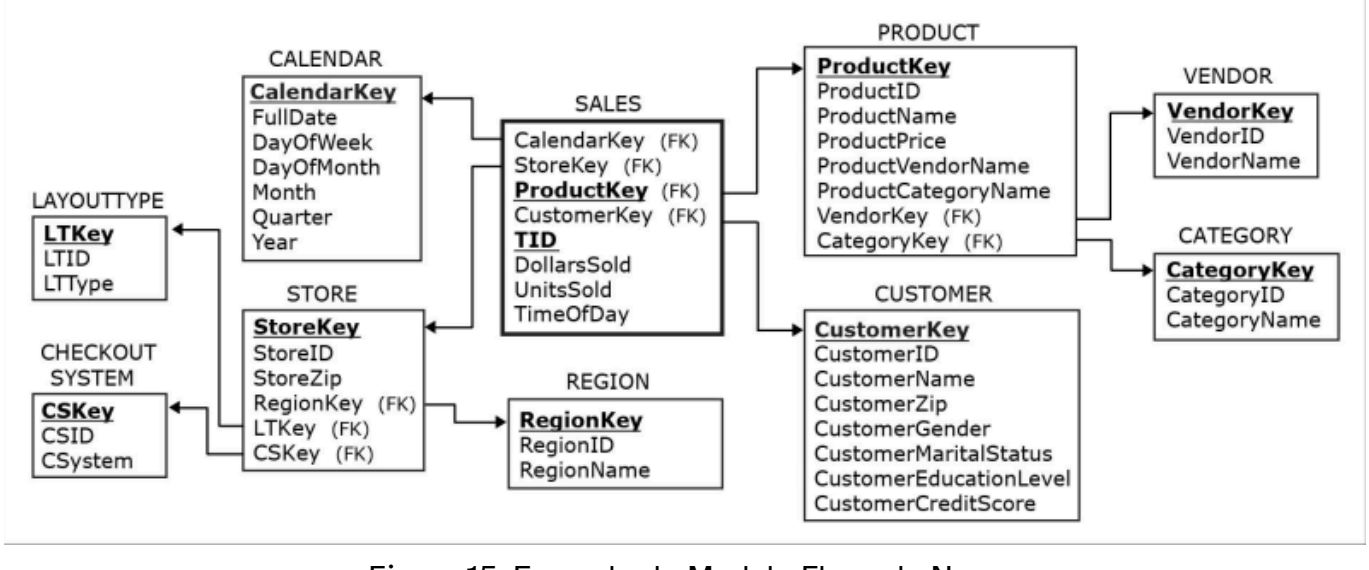


Figura 15: Exemplo do Modelo Floco de Neve

2.10 Abordagens de Datawarehouses

Vimos esse meio de se criar o Data Warehouse utilizando dos modelos estrela e floco de neve, porém, existem alguns métodos para abordar os Data Warehouses de formas diferentes. Porém, quando falamos de formas diferentes, não estamos dizendo que, por exemplo, se escolhermos uma dessas formas, automaticamente não podemos usar um modelo estrela, mas são algumas abordagens extras que podemos integrar dependendo do contexto

2.10.1 Data Warehouses Normalizados

Data warehouses utilizando de práticas-padrão de modelagem ER, de forma que ele mesmo serve como fonte para outros Data Warehouses que utilizam da modelagem dimensional e Data Marts dentro da empresa

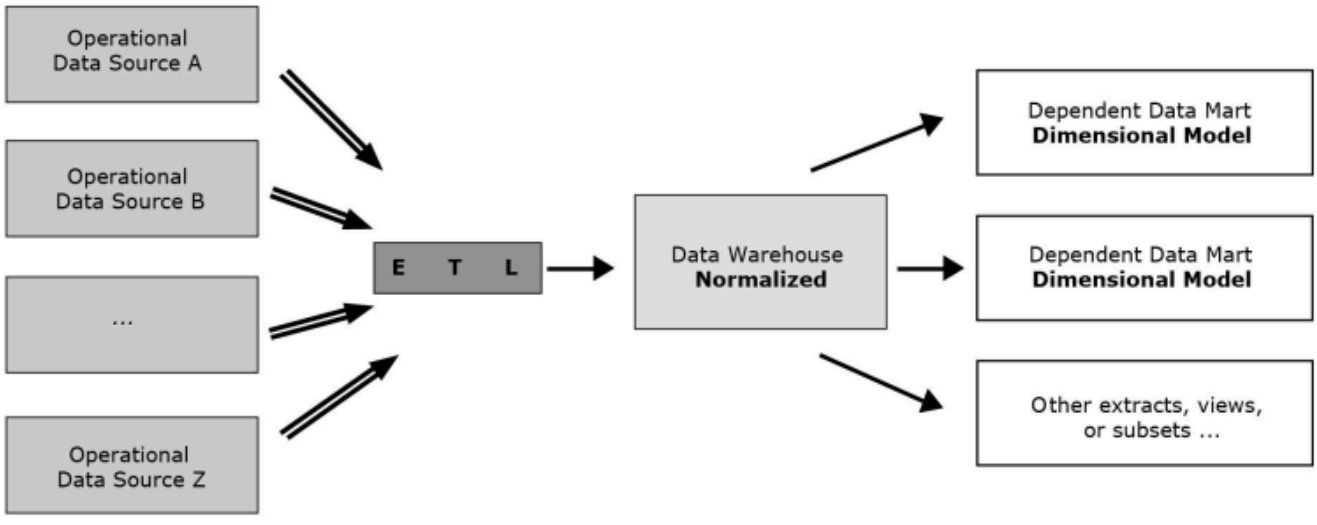


Figura 16: Data Warehouse Normalizado

2.10.2 Data Warehouse em Modelo Dimensional

Foi o que vimos na criação de Data Warehouses até esse momento, dividido em tabelas de dimensões e tabelas de fatos, onde um fato representa um acontecimento de interesse dentro do contexto do negócio e as dimensões são informações externas ao fato, mas que tem participação interna à ele

2.10.3 Data Marts Independentes

É quando vários **Data Marts** são criados em instâncias e setores diferentes da empresa, todos independentes um do outro. Consequentemente, isso faz com que **vários ETL's** sejam criados

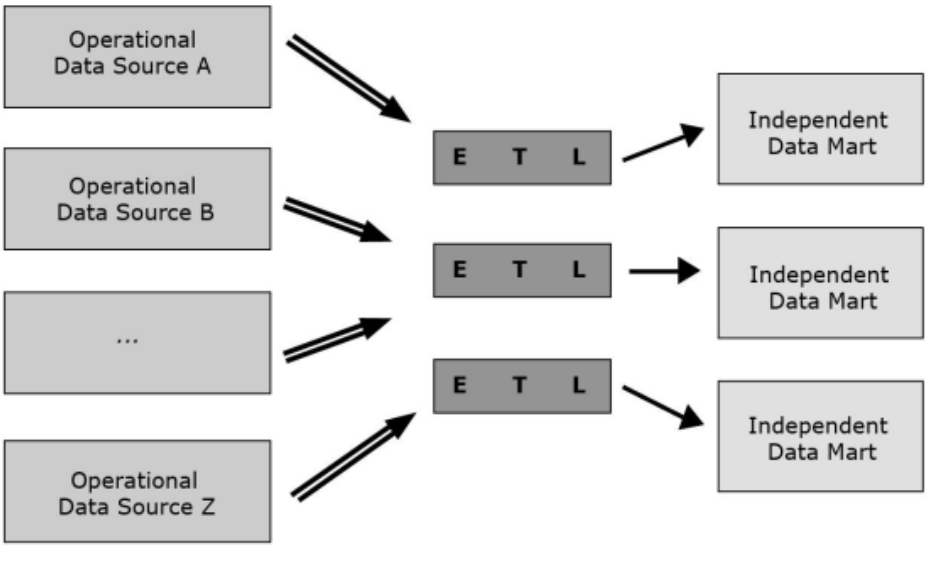


Figura 17: Data Marts Independentes

Essa abordagem é considerada inferior, já que inviabiliza uma análise **direto-ao-ponto** de toda a empresa e a existência de **vários** ETL's **não relacionados**

ETL e OLAP

Ainda vamos trabalhar dentro do contexto de **Data Warehouses**, porém, mesmo estando nesse escopo, **ETL** ainda merece um capítulo só para ele.

ETL é um script ou um cojunto de scripts com 3 objetivos principais:

3.1 Extrair

O ETL vai recuperar os dados analíticos úteis das devidas fontes e, eventualmente, serão carregados no Data Warehouse. O que vai ser extraído é **determinado na etapa de requisitos**

3.2 Transformação

Transformar a estrutura de dados coletada das fontes em estruturas compatíveis com o Data Warehouse modelado. O controle e melhoría da qualidade dos dados também estão inseridos nessa etapa.

3.2.1 Transformações Ativas

Após a transformação da estrutura de dados retornada pela **fonte**, a nova estrutura possui um número diferente de linhas (Quantidade de informações)

> *Exemplo*: Remoção de duplicatas, agregação de linhas, dimensões de mudança lenta (Tipo 2), etc.

3.2.2 Transformações Passivas

Após a transformação da estrutura de dados retornada pela **fonte**, a nova estrutura não possui alterações no número de linhas (Quantidade de informações)

> *Exemplo*: Gerar surrogate keys, atributos derivados, etc.

3.3 Load/Carga

Insere (Carrega) os novos dados de qualidade garantida no Data Warehouse. É um processo automatico que deve ser idealizado para que o usuário não tenha que se preocupar com o processo

Temos dois tipos de Load em um ETL:

> **Definição 3.3.1** (Carga Inicial): Preenche um Data Warehouse vazio. Pode envolver grandes quantidades de dados, dependendo de qual é o horizonte de tempo desejado dos dados no data warehouse recém-iniciado

> **Definição 3.3.2** (Carga de Atualização): Atualiza um Data Warehouse já iniciado. Feito em um período pré-determinado pela empresa (**Ciclo de Atualização**). Em **Data Warehouses ativos**, essas cargas ocorrem continuamente (Em microlotes)

3.4 Infraestrutura de um ETL

Normalmente, o processo de criação da infraestrutura ETL inclui o uso de ferramentas de software ETL especializadas e/ou código salvos. Devido à quantidade de detalhes que deve ser considerada, a criação de infraestrutura ETL é muitas vezes a parte que mais consome tempo e recursos no processo de desenvolvimento do data warehouse. Embora trabalhoso, o processo de criação da infraestrutura ETL é essencialmente predeterminado pelos resultados dos processos de coleta de requisitos e modelagem de data warehouse que especificam as fontes e o destino

3.5 Processamentos

Temos dois tipos de processamento de informações, na disciplina de Banco de Dados, analisamos os conceitos de OLTP, em Modelagem Informacional, estamos abordando a OLAP

Definição 3.5.1 (Online Transaction Processing (OLTP)): Atualização (Inserção, Remoção, Modificação), consultar e apresentar dados para fins operacionais

Definição 3.5.2 (Online Analytical Processing (OLAP)): Consultar e apresentar dados de Data Warehouses e/ou Data Marts para fins analíticos

e temos os chamados **OLAB / BI Tools** que são ferramentas que permitem os usuários fazer alterações estruturais de forma intuitiva em um Data Warehouse, como apenas com cliques no mouse. Basicamente um frontend para mexer num Data Warehouse diretamente