

Universidade Federal de Santa Catarina  
Centro de Blumenau  
Departamento de Engenharia de  
Controle e Automação e Computação



Humberto Zezulka Machado

Gerenciamentos de Sistemas de Execução Utilizando BI -  
Estudo de Caso

Blumenau  
2019

**Humberto Zezulka Machado**

# **Gerenciamentos de Sistemas de Execução Utilizando BI - Estudo de Caso**

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como parte dos requisitos necessários para a obtenção do Título de Engenheiro de Controle e Automação.  
Orientador: Prof. Dr. Mauri Ferrandin

Universidade Federal de Santa Catarina  
Centro de Blumenau  
Departamento de Engenharia de  
Controle e Automação e Computação

Blumenau  
2019

**Humberto Zezulka Machado**

# **Gerenciamentos de Sistemas de Execução Utilizando BI - Estudo de Caso**

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Engenheiro de Controle e Automação.

**Comissão Examinadora**

---

Prof. Dr. Mauri Ferrandin  
Universidade Federal de Santa Catarina  
Orientador

---

Prof. Dr. Marcos Vinicius Matsuo  
Universidade Federal de Santa Catarina

---

Prof. Dr. Daniel Martins de Lima  
Universidade Federal de Santa Catarina

Blumenau, 1 de fevereiro de 2019

Dedico este trabalho a todos aqueles que, de alguma forma,  
auxiliaram para a concretização desta etapa.

# Agradecimentos

A importância que dou para a conclusão desse ciclo é enorme e não tenho palavras para descrever a gratidão às pessoas que participaram dessa etapa. Primeiramente gostaria de agradecer a toda comunidade acadêmica envolvida na minha formação. Também gostaria de agradecer minha noiva, ela foi essencial nas grandes e pequenas escolhas tomadas nos últimos anos. Quero deixar registrado que reconheço o suporte que meu pai me ofereceu. Gostaria de agradecer minha irmã pelo suporte emocional, mas principalmente gostaria de agradecer minha mãe que durante minha vida inteira priorizou meus estudos, me ensinou a batalhar pelos meus objetivos e sempre esteve lá, na derrota e na vitória. "Nenhum homem é suficientemente sábio por si mesmo"(Titus Maccius Plautus).

*"You may never know what results come of your actions, but if you do nothing, there  
will be no results."  
(Mahatma Gandhi)*

# Resumo

A união de tecnologias é um processo natural e recorrente. Cada vez mais, ferramentas de coleta de dados são utilizadas nas indústrias com o objetivo de um controle maior sobre a linha de produção. Assim, uma enxurrada de dados é armazenada e a abundância dessa geração torna as análises necessárias demandarem muito tempo. A fim de facilitar a retirada de informações das grandes quantidades de dados, sistemas de *Business Intelligence* foram desenvolvidos e se utilizados da forma correta, podem exibir as informações mais relevantes de forma simples e intuitiva.

Esse trabalho retrata um cenário real de uma empresa que possui um sistema de coleta e controle de dados que atua sobre todas suas linhas de produção. Ainda apresenta uma solução de aprimoramento de um módulo do sistema que é aplicado apenas a uma linha da empresa, assim como a configuração e integração de uma ferramenta de BI para possibilitar a visualização das informações do sistema.

Os resultados dessa solução mostram como a generalização do módulo pode aumentar a abrangência dos sistemas da empresa, como a mudança de tecnologia alteram a performance do sistema, apresenta uma nova interface entre o usuário e o sistema, além do produto das aplicações de *Business Intelligence*, entre outros resultados.

Como conclusão desse trabalho é possível afirmar que dado os resultados esperados, as mudanças no módulo do sistema atenderam todas os requisitos especificados com algumas ressalvas, as aplicações de *Business Intelligence* apresentaram competência na extração informações de dados de uma linha de produção. Também é possível visualizar como os produtos das modificações e desenvolvimentos feitos nesse trabalho abrem portas para o crescimento do sistema.

**Palavras-Chave:** 1. MES. 2. *Business Intelligence* 3. *Business Analytics* 4. Inteligência de Negócios 5. Sistemas de execução de manufatura 6. Linha de Produção

# Abstract

The union between technologies is a natural and recurring process. Increasingly, data collection tools are used in industries for the purpose of increase control over the production line. Thus, a lot of data is stored and the abundance of that data generation makes the information analysis take too much time. In order to facilitate the extraction of information from large amounts of data, Business Intelligence systems have been developed and if used correctly, can display the most relevant information in a simple and intuitive way.

This project portrays a real scenario of a company that has a data collection and control system that operates on all its production lines. It also features a system module enhancement solution that is applied only to a company line, as well as the development of a BI to handle the large amount of the donated .

The results of this solution show how the generalization of the module can increase the comprehensiveness of the company's systems, also how the change of technology affect performance of the system, presents a new interface between the user and the system, as well as the product of Business Intelligence applications, among other results.

As conclusion of this work it is possible to state that given the expected results, changes in the system module have achieved all the objectives of requirements specified, Business Intelligence applications were proficient in extracting data information from a production line. It is also possible to visualize how the products of the modifications and developments made in this work open doors for the growth of the system.

**Keywords:** 1. MES. 2.*Business Intelligence* 3.*Business Analytics* 4.Manufacturing

Execution Systems 5. Production Line



# Lista de figuras

Figura 1 – Cenário Atual . . . . .	15
Figura 2 – Cenário Atual . . . . .	16
Figura 3 – Modelo Esquema Estrela . . . . .	22
Figura 4 – Linha de Produção e Grafos . . . . .	24
Figura 5 – Script Antigo - Funcionalidade única . . . . .	29
Figura 6 – Script Antigo - Parâmetros não variáveis . . . . .	29
Figura 7 – Exemplo Matriz Adjacência e Grafo. . . . .	33
Figura 8 – Fato Resultante . . . . .	36
Figura 9 – Ilustração Transformação . . . . .	38
Figura 10 – Segundo Passo Transformação . . . . .	39
Figura 11 – Terceiro Passo Transformação . . . . .	39
Figura 12 – Edição de Layout . . . . .	40
Figura 13 – Edição de Layout - Inserção de Componentes . . . . .	42
Figura 14 – Configuração Tasks . . . . .	42
Figura 15 – Interface de Parametrização . . . . .	45
Figura 16 – Interface de Parametrização de Layout . . . . .	46
Figura 17 – Script Novo - N linhas de produção . . . . .	46
Figura 18 – Script Novo - Parâmetros variáveis . . . . .	46
Figura 19 – Extração . . . . .	47
Figura 20 – Transformação . . . . .	48
Figura 21 – Carga . . . . .	48
Figura 22 – Esquema Resultante . . . . .	49
Figura 23 – <i>Dashboard</i> - Gráfico de Barras . . . . .	49
Figura 24 – <i>Dashboard</i> - Filtro de Data . . . . .	50
Figura 25 – <i>Dashboard</i> - Gráfico de Dispersão . . . . .	50
Figura 26 – <i>Dashboard</i> - Gráfico <i>SanKey</i> . . . . .	51
Figura 27 – Agendamento ETL . . . . .	52

# Lista de tabelas

Tabela 1 – Matriz de Adjacência da Figura 4 . . . . .	25
Tabela 2 – Comparação Performática . . . . .	45

# Lista de Siglas e Abreviaturas

BI	<i>Business Intelligence</i>
MES	<i>Manufacturing Execution System</i>
ETL	<i>Extract, Transform and Load</i>
QVD	<i>QlikView Data</i>
ERP	<i>Enterprise Resource Planning</i>
OLAP	<i>Online Analytical Processing</i>
DW	<i>Data Warehouse</i>
PDI	<i>Pentaho Schema Workbench</i>
API	<i>Application Programming Interface</i>
SAAS	<i>Software as a Service</i>
PAAS	<i>Platform as a Service</i>
RAM	<i>Random Access Memory</i>
AQL	<i>Acceptable Quality Limit</i>
FK	<i>Foreign Key</i>
PK	<i>Primary Key</i>
HTML	<i>Hypertext Markup Language</i>
XML	<i>Extensible Markup Language</i>
CSS	<i>Cascading Style Sheets</i>
MDX	<i>Multidimensional Expressions</i>
SQL	<i>Structured Query Language</i>
KPI	<i>Key Performance Indicator</i>
QMC	<i>Qlik Management Console</i>
UFSC	<i>Universidade Federal de Santa Catarina</i>

# Sumário

1	INTRODUÇÃO . . . . .	13
1.1	Problemática . . . . .	14
1.2	Objetivo . . . . .	15
1.3	Estrutura do Documento . . . . .	16
2	REVISÃO DE BIBLIOGRÁFICA . . . . .	17
2.1	Conceito de <i>Business Intelligence</i> . . . . .	17
2.1.1	A Evolução dos Sistemas de Informação de Negócios . . . . .	18
2.1.1.1	Primeira Geração: <i>Host-Based Query and Reporting</i> . . . . .	18
2.1.1.2	Segunda Geração: <i>Data Warehouse</i> . . . . .	18
2.1.1.3	Terceira geração: <i>Business Intelligence</i> . . . . .	18
2.1.2	Objetivos dos Sistemas de BI . . . . .	19
2.1.3	Ferramentas de BI disponíveis . . . . .	20
2.1.3.1	Pentaho . . . . .	20
2.1.3.2	Power BI . . . . .	20
2.1.3.3	QlikView . . . . .	20
2.1.3.4	Qlik Sense . . . . .	21
2.2	Arquitetura de BI . . . . .	21
2.2.1	Modelagem do Banco de Dados . . . . .	21
2.2.1.1	Tabela Fato . . . . .	22
2.2.1.2	Tabela Dimensão . . . . .	23
2.2.2	ETL . . . . .	23
2.3	Grafos e Matriz de Adjacência . . . . .	24
3	SISTEMA PROPOSTO . . . . .	26
3.1	Definição de Requisitos . . . . .	26
3.1.1	Contextualização do Problema . . . . .	26
3.1.2	Especificações do Problema . . . . .	27
3.2	Escolha das Ferramentas . . . . .	29
3.2.1	Interface de parametrização . . . . .	29
3.2.2	Linguagem e estrutura do algoritmo . . . . .	30
3.2.3	<i>Business Intelligence</i> . . . . .	30
3.3	Extração de Dados . . . . .	30
3.3.1	Extração de dados do algoritmo . . . . .	31
3.3.2	Extração <i>Business Intelligence</i> . . . . .	32

3.4	Implementações . . . . .	32
3.4.1	Melhorias Lógicas . . . . .	32
3.4.2	Parametrização - Dados Gerais . . . . .	32
3.4.3	Parametrização - Estrutura e Tempos de Transferência . . . . .	33
3.4.4	Modelagem dos <i>Dashboards</i> . . . . .	34
3.4.5	Modelagem de Dados . . . . .	35
3.4.6	ETL . . . . .	36
3.4.7	Criação do <i>Dashboard</i> . . . . .	40
3.4.8	Atualização Diária . . . . .	42
4	RESULTADOS . . . . .	44
4.1	Algoritmo . . . . .	44
4.1.1	Translação de Linguagem . . . . .	44
4.1.2	Performance . . . . .	44
4.1.3	Generalização . . . . .	45
4.2	<i>Business Intelligence</i> . . . . .	47
4.2.1	Extração . . . . .	47
4.2.2	Transformação . . . . .	47
4.2.3	Carga . . . . .	48
4.2.4	Atualização Diária . . . . .	51
5	CONCLUSÕES . . . . .	53
5.1	Trabalhos Futuros . . . . .	54
	REFERÊNCIAS BIBLIOGRÁFICAS . . . . .	55

# 1 Introdução

Por décadas empresas procuram por tecnologias para aumentar seu ganho de produtividade e, dessa forma, o mercado de sistemas de informação ligado à área de manufatura cresce de forma constante. Um número notável de grandes empresas adotaram sistemas como o *Enterprise Resource Planning* (ERP) e hoje empresas de pequeno e médio tamanho estão tomando a mesma medida. Contudo, durante o período de crescimento desses sistemas, os **especialista** em sistemas de informação não focaram no chão de fábrica [1]. Assim, não é incomum no presente o desenvolvimento de *softwares* de coleta de dados de produção usando banco de dados a fim de monitorar e controlar processos [2].

O conceito MES (*Manufacturing Execution System*) nasceu da procura de empresas de manufatura para atender as exigências dos mercados sob o ponto de vista da reatividade, qualidade, respeito aos padrões, redução de custos e prazos. Assim, as funções do MES são principalmente voltadas para atividades de manufatura. Historicamente, o sistema está consolidado em indústrias de processamento (indústrias farmacêuticas, alimentícias e de semicondutores), onde os sistemas MES atendem às necessidades de rastreabilidade impostas pelos supervisores, porém atualmente o sistema também é usado na maioria das indústrias de manufatura [2].

Em paralelo com o crescimento das **tecnologias de sistema**, a complexidade do ambiente de negócios no qual as organizações estão inseridas está aumentando. Desse modo, com o objetivo de não ser **engolida pelo mercado**, muitas empresas têm a necessidade de tomar decisões rápidas e com uma frequência maior. Tais decisões podem possuir natureza complexa e exigem uma análise profunda sobre dados obscuros.

Próximo dos anos de 1970, foram criados os primeiros sistemas de tomada de decisões relacionados a empresas. Naquela época, tais sistemas contavam com aplicativos operacionais como o de entrada de pedidos, controle de estoque e sistemas de folha de pagamento. Os anos se passaram e vários *softwares* de auxílio à decisão - informações administrativas, processamento analítico on-line (OLAP) e análise preditiva - surgiram e expandiram o domínio de suporte às decisões. Então, no começo da década de noventa, Howard Dressner, agregou o nome *Business Intelligence* (BI) para tais sistemas. **O BI é agora amplamente utilizado, especialmente no mundo da prática, para descrever aplicações analíticas** [3].

Neste trabalho será proposta uma solução para a generalização de uma ferramenta utilizada no sistema MES de uma empresa e a construção de uma aplicação de *Business Intelligence* que faça a interface do usuário com os dados retirados do sistema MES.

O projeto é pertencente a uma empresa real que preferiu que seu nome não fosse citado, assim como seus dados não fossem expostos. Dessa forma, além de não mencionar o nome da empresa, serão alterados dados como: nomes de campos de banco de dados, nomes de

tabelas, nomes de arquivos, qualquer valor exposto e qualquer código fonte. Todavia, os conceitos e lógicas utilizados nesse trabalho serão mantidos.

## 1.1 Problemática

Atualmente a empresa em questão conta com inúmeras linhas de produção espalhadas pelo mundo e todas elas possuem um sistema MES implementado que tem como principal função armazenar o estado de trabalho de cada estação (operante, bloqueada, falta de peça, entre outras informações). Esses dados são armazenados em um banco de dados e por se tratar de uma linha de produção com muitas estações a quantidade de dados gerados é alta.

Em uma das linhas de produção existe acoplado ao sistema MES um algoritmo que tem como principal função o rastreamento de ocorrências específicas - é denominado ocorrência quando uma estação entra em qualquer modo diferente do operante. Dessa forma, essa ferramenta consegue descobrir o motivo de uma estação ter entrado em um estado que seja diferente do estado operante. Contudo, a ferramenta foi desenvolvida para apenas uma linha de produção, que foi designada pela empresa. Todos os parâmetros, como *layout* da linha de produção, estados a serem considerados pela ferramenta, entre outros estão fixados para uma linha de produção específica. A Figura 1 ilustra a situação atual do sistema.

Atualmente os supervisores da área que administram as linhas de produção da empresa requisitaram que tal ferramenta fosse implantada em todas as linhas de produção, visto que todas possuem o sistema MES implantado. Porém, o fato da ferramenta estar com seus parâmetros fixos para apenas uma linha de produção, impossibilita essa requisição dos supervisores, surgindo assim, um empecilho para a empresa.

Ainda que ocorra a implementação da ferramenta em todas as linhas de produção da empresa, todos os dados vão estar presentes apenas no banco de dados. Dado que, a interface do banco de dados com o usuário não é o ambiente mais apropriado para tirar informações estatísticas, foi gerada a necessidade de um outro meio para visualizar essas informações.

Ao analisar as ferramentas disponíveis no mercado os supervisores concordaram que uma aplicação de *Business Intelligence* atenderiam suas necessidades. Assim, surgiu outra etapa do projeto: o desenvolvimento de uma ferramenta de BI.

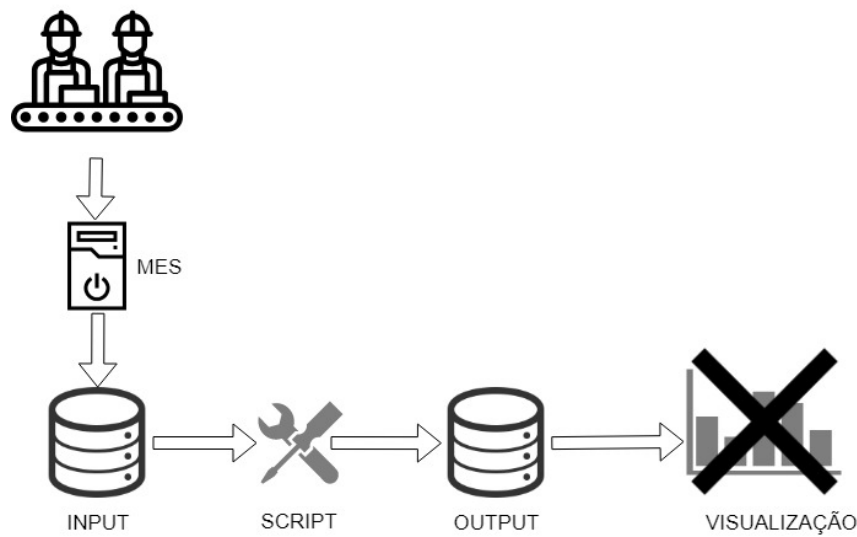


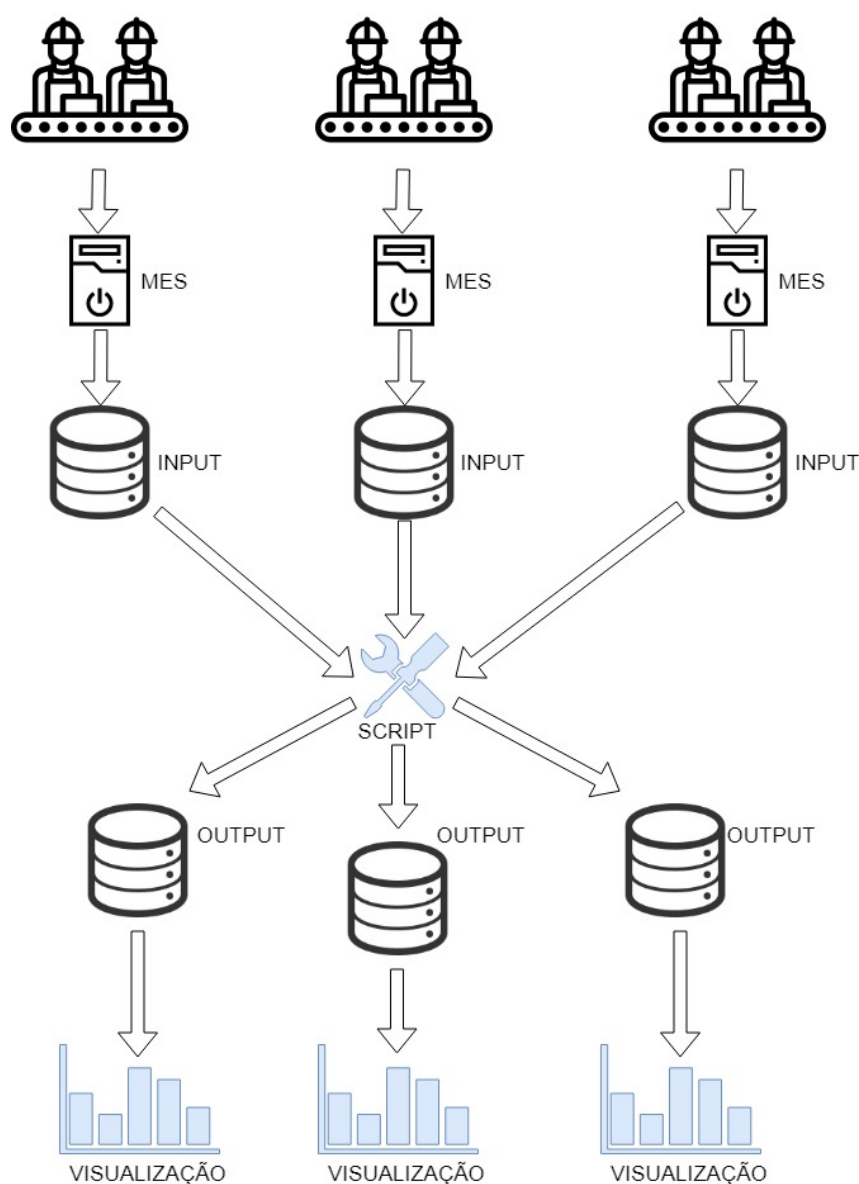
Figura 1 – Cenário Atual

## 1.2 Objetivo

Esse projeto tem como objetivo propor uma solução para tornar a ferramenta acoplada ao sistema MES, que se encontra em apenas uma linha de produção, aplicável a qualquer linha de produção da empresa. Para isso, será feita a implementação de um sistema de *Business Intelligence* na empresa que apresente informações do resultado da ferramenta.

Como premissa do usuário, as modificações realizadas na atual ferramenta não podem alterar a lógica existente e deve haver a possibilidade de interação entre um usuário leigo e os parâmetros a serem modificados na implementação de uma nova linha. Quanto ao sistema de *Business Intelligence*, foi requisitado que determinados indicadores e filtros aparecessem na aplicação, mas os gráficos que mostrariam o resultado poderiam ser sugeridos pelo desenvolvedor. A Figura 2 ilustra o resultado esperado do sistema.



Figura 2 – **Cenário Atual**

### 1.3 Estrutura do Documento

Este documento é dividido em cinco capítulos. O segundo capítulo apresenta os embasamentos teóricos que sustentaram as soluções propostas e as ferramentas utilizadas. O capítulo três apresenta um aprofundamento técnico da problemática e o detalhamento das soluções propostas. O quarto capítulo apresenta os resultados das soluções e o último capítulo apresenta a conclusão e possíveis trabalhos futuros.

## 2 Revisão de Bibliográfica

Essa seção irá trazer os conceitos relacionados à *Business Intelligence* para melhor entendimento do projeto que será apresentado.

### 2.1 Conceito de *Business Intelligence*

A cada dia o mercado se torna mais competitivo e por isso as empresas estão sempre buscando formas de analisar os seus negócios. A necessidade da informação rápida e eficiente passa a ser indispensável. Uma solução para obter essas informações é a utilização de sistemas de Inteligência de Negócios - *Business Intelligence*, que auxiliam na tomada de decisão estratégica através do fornecimento das informações necessárias de forma rápida e concisa [4].

De acordo com o livro "*Getting Started with Data Warehouse and Business Intelligence*", de Almeida et. al, *Business Intelligence* (BI) significa utilizar os dados ativos para aprimorar as decisões de negócios. Se refere à acesso, análise e descoberta de novas oportunidades [5].

Os sistemas de BI apresentam três níveis de estrutura tecnológica. O nível inferior faz a integração entre o servidor de *Data Warehouse* (DW) e as aplicações de suporte ao processo de *Extract, Transformation and Loading* (ETL). O nível intermediário faz a integração do servidor de *Online Analytical Processing* (OLAP). Por fim, o nível superior integra as ferramentas de análise e de fornecimento de informação relevante resultante [6].

Grande parte dos sistemas de BI utiliza ferramentas de cubos de dados. Esses cubos de dados são chamados de *Online Analytical Processing* (OLAP) e são implementados em banco de dados multidimensionais. Os dados são armazenados e indexados pelos cubos em formatos dimensionais. Assim, pesquisar dados em um cubo OLAP se torna rápido e eficiente, já que o cubo utiliza pré-cálculos, estratégias de indexação e outras otimizações [7].

Para obter as informações desejadas pela gerência da empresa, o sistema de BI acessa o cubo de dados e realiza as pesquisas necessárias. Com isso, é possível criar as visualizações, que representam os dados de forma gráfica. Dentro do sistema de BI, os dados conseguem representar objetos abstratos, como lucro, vendas ou custo. As visualizações são usadas para criar painéis chamados de *Dashboards*, que traz várias informações em formas de gráficos e tabelas [8].

De acordo com Turban et al. (2009), os *Dashboards* geram uma visão ampla e visual dos dados com os indicadores-chave de desempenho, exibindo tendências e exceções do desempenho corporativo [9]. Com os *Dashboards*, a ação do tomador de decisão para

o administrador se torna mais fácil, visto que o mesmo utiliza os sistemas de BI para direcionar as ações estratégicas da empresa [10].

### 2.1.1 A Evolução dos Sistemas de Informação de Negócios

Segundo o livro "*Getting Started with Data Warehouse and Business Intelligence*", de Almeida et. al, os sistemas de informação de negócios são divididos em três gerações: *Host-Based Query and Reporting*, *Data Warehousing* e *Business Intelligence*, que são serão descritas a seguir [5].

#### 2.1.1.1 Primeira Geração: *Host-Based Query and Reporting*

Os primeiros sistemas de informação de negócios utilizavam aplicações para fornecer as informações que os usuários necessitavam. Os resultados dessas aplicações normalmente vinham em grande volume de papel e os usuários precisavam percorrer todas as informações até encontrar a resposta desejada para determinada questão do negócio [5].

Essa primeira geração de sistemas de informações de negócios só era usada por quem tinha profundo conhecimento dos dados e muita experiência com computadores, como Analistas de Negócios. Os executivos e gerentes de negócios precisavam confiar nos analistas para responder as perguntas e para fornecer-lhes as informações que precisavam [5].

#### 2.1.1.2 Segunda Geração: *Data Warehouse*

A segunda geração dos sistemas de informações utilizava os **armazenamentos de dados**, conhecidos como *Data Warehouses*, que aumentou muito a **capacidade**. Eles tinham várias vantagens sobre a primeira geração [5].

Uma das vantagens é que os DW's são projetados para atender as necessidades dos usuários de negócio, e não do operacional. Outra vantagem é que as informações são limpas e consistentes e são armazenadas de uma forma que os usuários podem entender [5].

Os DW's conseguem fornecer, além das informações atuais, um histórico e um resumo das informações. Além disso, podem ser usados com interfaces aprimoradas e fornecem ferramentas de tomadas de decisão eficazes [5].

#### 2.1.1.3 Terceira geração: *Business Intelligence*

Apesar de a segunda geração já ter dado um grande salto em relação a primeira, ela ainda tinha como principal objetivo o armazenamento de dados. A terceira geração veio então com um conceito diferente, onde as atenções estão viradas para ferramentas que auxiliem os usuários na tomada de decisão, facilitando o acesso aos dados [5].

Os sistemas de BI concentram em melhorar o acesso e a entrega de informações. Para isso, eles utilizam sistemas de processamento analítico online (OLAP) e ferramentas gráficas avançadas. Portanto, os sistemas de BI devem fornecer escalabilidade e serem capazes de suportar e integrar diferentes produtos [5].

Um dos principais objetivos do sistema de BI é facilitar o acesso aos dados, fazendo com que os usuários passem a acessar menos os bancos de dados. As informações que precisam para tomar decisões de forma rápida, podem ser obtidas com o acesso às aplicações ou recebendo-as em um intervalo predefinido por meio de uma intranet corporativa ou por e-mail [5].

### 2.1.2 Objetivos dos Sistemas de BI

O sistema de BI precisa atender alguns requisitos para que sua eficácia seja atendida, dentre elas estão:

- **O sistema de BI deve tornar as informações facilmente acessíveis.**

O entendimento do conteúdo apresentado deve ser fácil e os dados devem ser intuitivos para os usuários. As ferramentas precisam ser simples e fáceis de utilizar. Além disso, as informações devem ser trazidas no menor tempo possível, ou seja, o sistema precisa ser rápido e simples [7].

- **O sistema de BI deve apresentar informações de forma consistente.**

O sistema deve ser confiável e quando as informações vierem de fontes diferentes devem ser cuidadosamente agregadas. Somente deve ser entregue ao usuário final quando estiver com qualidade e pronto para ser usado [7].

- **O sistema de BI deve apresentar as informações a tempo.**

Os dados gerados devem ser convertidos e disponibilizados para os usuários em um curto período de tempo, já que os sistemas de BI são utilizados de forma muito intensa para as decisões operacionais [7].

- **O sistema de BI deve servir como base competente e confiável para tomadas de decisão.**

O sistema deve apresentar os dados corretos para auxiliar nas tomadas de decisão. A funcionalidade mais importante de um sistema de BI é apresentar análises importantes das informações necessárias de forma confiável [7].

- **O sistema de BI deve ser adaptável às mudanças.**

A tecnologia, as condições de negócios, os dados e as necessidades dos usuários estão suscetíveis à mudanças. Por isso, o sistema de BI deve ser desenvolvido para adaptar-se a essas mudanças sem que dados ou aplicações sejam invalidados [7].

- **O sistema de BI deve proteger as informações disponíveis.**

Muitas informações valiosas das empresas são guardadas nos DW's. Sendo assim, é muito importante que essas informações se mantenham protegidas para que não sejam acessas por pessoas não autorizadas [7].

### 2.1.3 Ferramentas de BI disponíveis

A seguir serão apresentadas as ferramentas de BI disponíveis no mercado hoje que são mais utilizadas pelas empresas.

#### 2.1.3.1 Pentaho

Pentaho é um software de BI de código aberto que foi desenvolvido em Java pela Pentaho Corporation. É possível desenvolver um sistema de BI utilizando o Pentaho, incluindo processos de extração de dados, criação de cubo de dados, criação de *Dashboards*, entre outras várias funcionalidades. Para utilizá-lo, é necessário que se tenha um conhecimento básico de programação nas linguagens JavaScript, HTML, CSS e MDX [11].

O Pentaho possui diversos componentes. Dentre eles, os principais são: *Pentaho Data Integration* (PDI), que é a ferramenta de extração, transformação e carga de dados; *Pentaho Schema Workbench*, que é uma ferramenta de criação de cubo OLAP; e *Pentaho BI Server*, que é a ferramenta de integração com o usuário [11].

#### 2.1.3.2 Power BI

Power BI é um software de BI criado pela Microsoft. Pode ser utilizado tanto para dados simples, quanto para dados em nível empresarial. Se trata de um BI *self-service*, visto que é de fácil manuseio e veloz em transformar a interação entre os dados em visualizações dinâmicas. Ele conta com o Power BI Desktop, que é uma aplicação do Windows, e também com o serviço Power BI (SaaS), que é um serviço online [12] [10].

Esse sistema também conta com o Power BI Embedded, que simplifica as funcionalidades do Power BI, permitindo a criação rápida de visuais, relatórios e *Dashboards*. O serviço do Power BI (SaaS) e o serviço do Power BI Embedded no Azure (PaaS) têm APIs para inserir seus painéis e relatórios. Com isso é possível ter acesso aos recursos mais recentes do Power BI, como painéis, *gateways* e espaços de trabalhos de aplicativos [12].

#### 2.1.3.3 QlikView

QlikView é um software de BI desenvolvido pela empresa Qlik. Ele se concentra na descoberta de negócios pelo usuário final através de visualizações e exploração de dados.

Para ser utilizado nas empresas é necessário adquirir uma licença para cada usuário, o que ocasiona no aumento dos custos [13].

A principal vantagem da ferramenta é a alocação dos dados em memória RAM. Dessa forma, o QlikView utiliza uma tecnologia chamada de AQL no lugar da OLAP. Essa tecnologia utiliza uma tabela Fato que é associada a todas as outras tabelas. Tais atribuições possibilitam um aumento da velocidade de resposta do sistema e uma diminuição no tempo do projeto [13].

#### 2.1.3.4 Qlik Sense

O Qlik Sense também é um software de BI criado pela Qlik. Pode ser usado para assuntos pessoais e para empresas de qualquer porte. Ele é uma variação do QlikView e utiliza a mesma tecnologia. Focado no auto-serviço, ele permite criar rapidamente visualizações, explorar dados e oportunidades em todos os ângulos [14].

Um dos diferenciais do Qlik Sense é que se trata de um software responsivo, ou seja, é possível acessá-lo pelo computador, *tablet* ou *smartphone* e ele se adaptará automaticamente. Outro diferencial é o serviço Qlik Sense Cloud, onde é possível compartilhar aplicativos do Qlik Sense na nuvem de forma gratuita [14].

Para o desenvolvimento desse projeto, optou-se por utilizar o Qlik Sense, visto que é uma ferramenta que a empresa já possui uma licença e que é mais dinâmica e intuitiva para o desenvolvedor e para os usuários.

## 2.2 Arquitetura de BI

Nesta seção serão apresentados conceitos sobre Modelagem de Banco de Dados e ETL, que são fundamentais para o desenvolvimento do projeto.

### 2.2.1 Modelagem do Banco de Dados

A modelagem dimensional tem o objetivo de organizar o banco de dados de forma simples, de modo que os *softwares* consigam buscar as repostas de forma mais rápida e eficiente. Essa organização também permite que os usuários entendam as informações mais facilmente [7].

O modelo mais utilizado na modelagem dimensional é o Esquema Estrela ou *Star Schema*, que melhora a performance dos sistemas de consulta e dá suporte às tomadas de decisão. A principal característica desse modelo é a presença de dados altamente redundantes, o que melhora o desempenho do sistema [7].

O modelo Esquema Estrela possui dois componentes principais, que são a Tabela Fato e as Tabelas Dimensão e é esquematizado como na Figura 3.

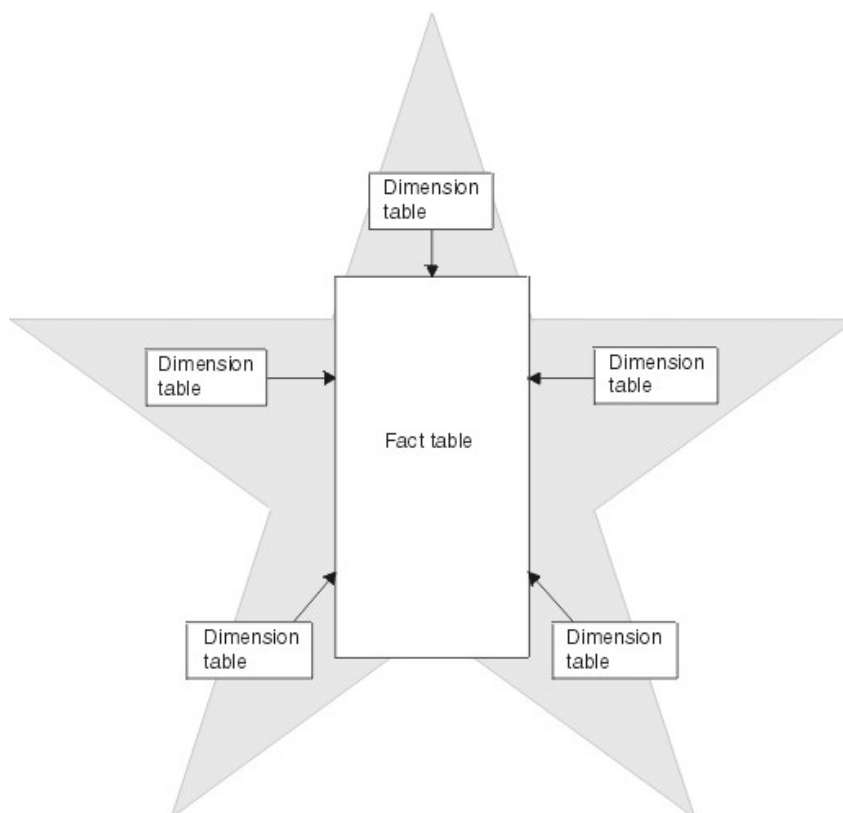


Figura 3 – Modelo Esquema Estrela

#### 2.2.1.1 Tabela Fato

A Tabela Fato tem a função de armazenar as medidas de desempenho que foram resultadas de eventos de processos de negócios de uma empresa. O termo "Fato" representa uma medida de negócio [7][15].

Na Tabela Fato, cada linha corresponde a um evento de medição. Cada linha contém dados que estão em um nível de detalhamento específico, chamado de grão, que pode ser, por exemplo, um produto vendido em uma determinada transação. Uma das premissas mais importantes para a criação da tabela Fato é deixar todas as linhas no mesmo grão, garantindo assim, que a medida não se repita ao longo da tabela [15][16][7].

Geralmente, a tabela Fato contém dados numéricos e aditivos, como o valor de um produto em reais. Muitas vezes, os sistemas de BI trazem centenas ou milhares de linhas, sendo muito útil poder realizar a soma dos dados, por isso a aditividade é uma propriedade muito importante [7][16][6].

Todas as tabelas Fato contém pelo menos duas ou mais Chaves Estrangeiras (*Foreign Key* - FK), que conectam às chaves primárias das tabelas Dimensão. Quando todas as chaves na tabela Fato correspondem corretamente às respectivas chaves primárias das tabelas Dimensão, essas tabelas satisfazem uma integridade referencial. Assim, é possível acessar a tabela Fato através das tabelas Dimensão associadas à ela [7].

### 2.2.1.2 Tabela Dimensão

As tabelas Dimensão são complementares à tabela Fato. Elas compreendem os valores textuais que estão associados aos eventos contidos na tabela Fato. Esses valores descrevem "quem, o que, quando, onde, como e por que" associados a cada evento. É comum que as tabelas Dimensão possuam um grande número de colunas ou atributos. Uma única Chave Primária (*Primary Key* - PK) define cada dimensão e funciona como referência para a tabela Fato [7][16].

Os atributos de dimensão servem como fonte primária de restrições de consulta, agrupamento e rótulos de relatório. Em uma solicitação de consulta ou relatório, os atributos são identificados como palavras. Um exemplo prático é quando um usuário deseja visualizar as vendas em reais por marca, essa marca deve ser um atributo de dimensão [15].

Os atributos da tabela Dimensão exercem uma função fundamental nos sistemas de DW/BI, já que representam a fonte de praticamente todas as restrições e rótulos de relatórios. Eles devem consistir em palavras reais e as abreviações devem ser evitadas ao máximo, fornecendo consultas mais consistentes. Os atributos são essenciais para tornar os sistemas DW/BI utilizáveis e compreensíveis [15][16].

Certas informações podem aparecer de forma redundante nas tabelas Dimensão, porém, não se trata de um problema e não deve ser feita a normalização dessas informações em outra tabela. Como as tabelas Dimensão são menores que a tabela Fato, não é vantagem normalizar as informações que estão contidas nelas, sendo mais interessante manter a simplicidade e acessibilidade [7].

## 2.2.2 ETL

A sigla ETL se trata de um processo de Extração, Transformação e Carga (*Extract, Transform and Load*), que é a parte do sistema de BI encontrada entre os sistemas operacionais de origem e a área de apresentação do BI. É um processo que extrai dados de um sistema de Base de Dados, processa e modifica esses dados, e em seguida insere em uma outra Base de Dados [17].

A primeira parte do processo é a Extração, onde os dados para o *Data Warehouse* são obtidos. Extração significa ler e entender os dados de origem e copiar os dados necessários no sistema ETL para manipulação adicional. A partir daí, os dados passam a pertencer ao *Data Warehouse* [18]. Os dados podem ser obtidos de diferentes fontes, como bancos de dados, arquivos de texto, documentos HTML ou XML, arquivos .xls, .csv, entre outros [19].

Geralmente, os dados extraídos são armazenados em um banco de dados relacional, facilitando o processamento de dados posteriormente na etapa da transformação. O soft-



ware que realiza a extração registra informações de tempo de extração, estrutura dos dados e local de origem dos dados [19] [18].

Após a Extração das informações desejadas, tem-se a etapa da Transformação dos dados obtidos. Essa etapa é considerada a mais complexa do processo de ETL. A transformação de dados se trata da unificação de dados, cálculo de **agregados** necessários, identificação de dados perdidos ou duplicação dos dados, ou seja, é responsável por realizar alterações necessárias nas informações para que fiquem de acordo com a necessidade do sistema de BI [19][7].

A última parte do processo é a Carga, que carrega as informações com suas alterações feitas para o *Data Warehouse*. É neste passo que os dados são inseridos em suas respectivas tabelas Fato e Dimensão, que serão posteriormente utilizadas no sistema de BI [7]. Quando as tabelas Fato e Dimensão do modelo dimensional forem atualizadas, indexadas, com qualidade assegurada e com os dados apropriados, os usuários são notificados de que novos dados foram publicados [19] [18].

## 2.3 Grafos e Matriz de Adjacência

Segundo (Ayhan & Wortman, 1999) grafos são bem adaptados para representar uma classe de estrutura de manufaturas que apresentem apenas atrasos e fenômenos de sincronização [20].

Um grafo simples é uma estrutura discreta formada por um conjunto não vazio de vértices  $V$  e um conjunto de arestas  $A$ . Cada aresta é formada por um par de vértices distintos e para cada par de vértice existe no máximo uma aresta associada [21].

A Figura 4 exemplifica como uma linha de produção pode ser representada em forma de grafos.

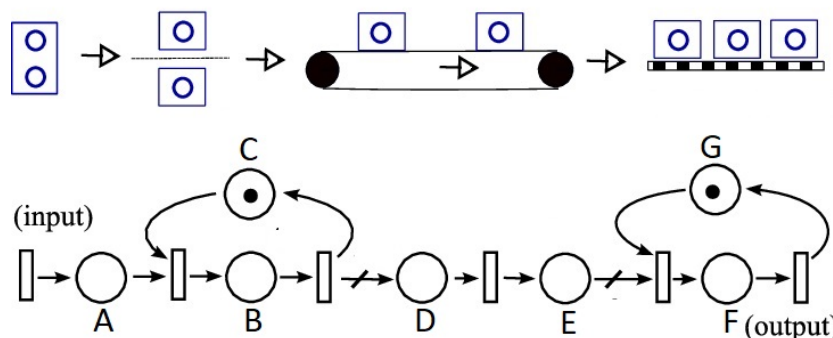


Figura 4 – Linha de Produção e Grafos

Um grafo  $G = (V, A)$ , com  $|V| = n$  e  $|A| = m$ , pode ser descrito por uma matriz de adjacência. A matriz de adjacência é uma matriz  $M = m_{i,j}$  simétrica  $n \rightarrow n$  que armazena

o relacionamento entre os vértices do grafo. Cada entrada  $m_{i,j}$  é igual a [21]:

$$m = \left\{ \begin{array}{l} 0, \text{ se } i \text{ não é adjacente a } j \\ m, \text{ onde } m \text{ é a quantidade de arestas incidentes tanto em } i \text{ quanto em } j \text{ (com } i \neq j\text{)}. \\ p, \text{ onde } p \text{ é a quantidade de laços incidentes em } i = j. \end{array} \right\}$$

A matriz de adjacência da Figura 4 poderia ser descrita como:

	A	B	C	D	E	F	G
A	0	1	0	0	0	0	0
B	1	0	1	1	0	0	0
C	1	0	0	0	0	0	0
D	0	1	0	0	1	0	0
E	0	0	0	1	0	1	0
F	0	0	0	0	1	0	1
G	0	0	0	0	0	1	0

Tabela 1 – Matriz de Adjacência da Figura 4


## 3 Sistema Proposto

Este capítulo tem como objetivo descrever o modo no qual o sistema desenvolvido e os detalhes de sua implementação. Para melhor entendimento serão abordados os detalhes do problema para então apresentar os tópicos de escolhas das ferramentas, extração de dados e por fim a implementação.

### 3.1 Definição de Requisitos

Nesta seção será descrita a contextualização dos problemas e quais requisitos e métodos serão usados para resolução dos problemas.

#### 3.1.1 Contextualização do Problema



Hoje, todas as linhas de produção da empresa possuem um sistema de sensores que armazena o estado funcional das estações a cada mudança de estado. Este armazenamento é realizado em um banco de dados - cada linha de produção possui o seu - e cada registro do banco representa uma ocorrência. Os campos salvos em cada ocorrência são os seguintes: identificação da ocorrência, identificação da linha de produção, identificação da estação, início da ocorrência, fim da ocorrência, duração da ocorrência, identificação da causa da ocorrência, descrição da causa e *flag* de tratamento (o campo só terá um valor quando a ocorrência for tratada). O sistema de armazenagem roda em tempo real, ele não depende da linha de produção para funcionar mas está programado para atuar no mesmo período de funcionamento da linha.

Especificamente em uma das linhas de produção da empresa existe um algoritmo independente que é executado três vezes por dia e trata algumas ocorrências da linha que possuem como causa uma parada desta linha. Dado uma estação, o propósito do algoritmo é identificar quais estações causaram paradas que refletiram na estação em questão, além de identificar as causas da ocorrência. Esse algoritmo não é compatível com outras linhas como mostrado na Figura 5.

Atualmente, esse sistema tem como parâmetros de entrada a estação a ser tratada (também chamada de estação gargalo) e parâmetros do banco de dados. Já a saída do sistema é uma tabela que contém as seguintes informações: identificação da ocorrência tratada, identificação da ocorrência causadora, identificação da estação tratada, início da ocorrência, fim da ocorrência, duração e percentual.

Essas informações podem ser descritas como: O campo identificação da ocorrência tratada é referente a cada ocorrência que estava salva e foi tratada pelo sistema; o campo identificação da ocorrência causadora contém a informação de qual ocorrência produziu a

ocorrência na estação gargalo; o campo identificação da estação tratada se refere a estação gargalo; a coluna início da ocorrência salva a data original com precisão em segundos da ocorrência causadora; a mesma lógica se aplica na coluna fim da ocorrência; o campo duração é dado em segundos e é a subtração do campo fim da ocorrência menos início da ocorrência; o campo percentual é referente a quanto da duração dessa ocorrência causadora representa da duração da ocorrência original.

A tabela contendo essas informações é salva incrementalmente no mesmo banco de dados que contém a tabela original. Atualmente, a empresa possui dois problemas: (i) tornar o algoritmo aplicável a mais linhas de produção; e (ii) implantar uma ferramenta que facilite a visualização dos resultados do algoritmo em questão.

### 3.1.2 Especificações do Problema

Os problemas a serem resolvidos podem ser divididos em dois grupos. Tornar o algoritmo aplicável a qualquer linha de produção da empresa e o desenvolvimento de uma aplicação de *Business Intelligence*.

Quanto ao primeiro problema é possível decompor em quatro partes: parâmetros gerais, estrutura, tempos de transferência e causas de interrupções.

- Parâmetros gerais

Atualmente, o algoritmo possui os seguintes parâmetros do banco fixados: nome, domínio, instância, login, senha, tabelas e colunas. Além dos parâmetros do banco ainda é fixa a identificação da estação gargalo, limite de linhas a ser buscada no banco, data inicial que as buscas devem ocorrer, duração da interrupção e códigos de causas.

Como pré-requisito do sistema, todos esses parâmetros devem se transformar em variáveis do sistema para que seja possível a conexão com novos bancos, definições de novas estações gargalos e a mudança dos parâmetros essenciais para o algoritmo.

- Estrutura

A disposição das estações é importante dentro do algoritmo, pois a mesma define qual estação é anterior ou a seguinte. Logo, qualquer mudança física da linha deve ser corrigida na estrutura digital do *script*. Dessa forma, o algoritmo existente possui sua definição em duas estruturas de *switch-case* [22]. A primeira estrutura retorna a estação ou estações imediatas seguintes e o parâmetro de cada caso é definido pela descrição da estação. A segunda estrutura de *case* também é definida da mesma forma, contudo retorna a estação ou estações anteriores a referente.

É uma **requisição** que no novo sistema a estrutura seja facilmente mutável sem mudanças no código fonte.

- Tempos de Transferência

O tempo que uma interrupção leva para se propagar entre estações é um parâmetro importante na lógica do algoritmo pois é ele que determina o período que as ocorrências devem estar. Este tempo varia de estação para estação, visto cada uma tem um tempo variável de atuação. Porém, atualmente esse parâmetro é fixado e contém só dois valores: Tempo de transferência de avanço e de recuo.

Outro pré-requisito é que os tempos sejam variáveis de acordo com a estação, alterável e inseríveis como entrada do sistema.

- Causas de Interrupções

As interrupções podem ser classificadas em três diferentes categorias: bloqueio, falta de peça e falha genérica. Dentro de cada categoria podem existir diversas subcategorias definidas pelo usuário da ferramenta. Todas as ocorrências possuem uma subcategoria diferente, e caso o usuário deseje que a subcategoria de interrupção seja tratada é importante que a mesma seja classificada dentro das três categorias e especificada no algoritmo.

A distinção de cada categoria é dada pela direção da propagação da falha. Na classe bloqueio, as estações que estão a frente da mesma em que ocorreu a falha vão parar de funcionar, logo a propagação é na direção positiva da linha. Já na classe de falta de peça, a propagação é no sentido negativo e as estações anteriores vão entrar em pausa. Por fim, as falhas de natureza ou falhas genéricas não têm direção de propagação, ou seja, a linha para por inteiro ao mesmo tempo.

Deve ser projetado um sistema onde o usuário pode escolher as causas e em quais categorias elas devem ser classificadas. No código existente as subclasses não são parâmetros do sistema, encontrando-se fixas no código fonte, logo, essa ferramenta só funciona para uma linha de produção.

Todos os parâmetros citados acima são estáticos dentro do algoritmo, como ilustra a Figura 6. É uma requisição que o novo sistema possibilite uma mudança de tais parâmetros facilmente.

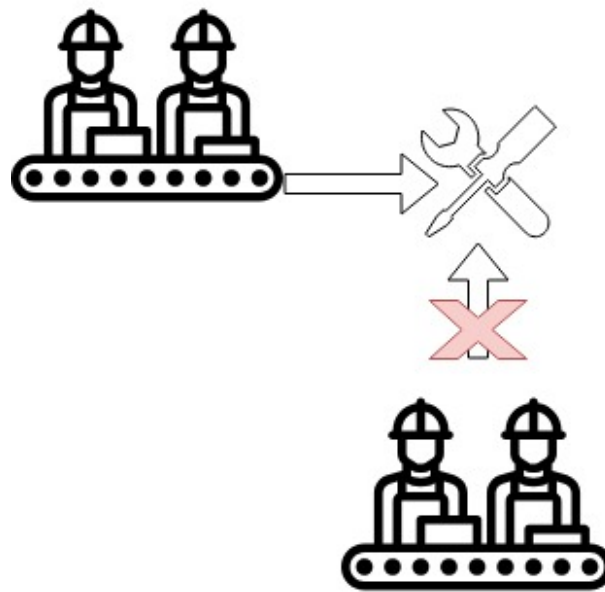


Figura 5 – Script Antigo - Funcionalidade única

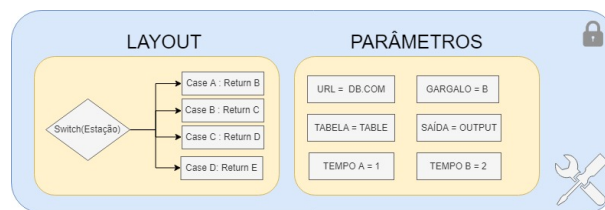


Figura 6 – Script Antigo - Parâmetros não variáveis

- *Business Intelligence*

A análise dos resultados e visualização de dados deve se dar por uma ferramenta de *Business Intelligence* contendo informações especificadas pelo usuário.

## 3.2 Escolha das Ferramentas

A solução proposta necessita de três ferramentas. A primeira será utilizada na interface com o usuário com o objetivo de parametrização do algoritmo, a segunda é a linguagem e estrutura do algoritmo utilizado e por fim a ferramenta de *Business Intelligence*.

### 3.2.1 Interface de parametrização

Sabido que existem diversos parâmetros que devem ser determinados pelo usuário e os mesmos podem ser alterados quando for preciso, faz-se necessário a criação de uma interface com o usuário para que o mesmo possa alterar ou inserir tais informações. Determinou-se então, que a interface utilizada será um arquivo Excel.

A escolha do arquivo Excel foi devido à familiaridade que os analistas que irão acessar ao sistema têm com a ferramenta. Determinados os parâmetros que podem ser alterados, a planilha deve conter colunas com os títulos dos campos correspondentes. Assim o usuário deve apenas preencher a planilha de acordo com o título do campo. Além dos parâmetros de banco de dados, causa de interrupções e parâmetros gerais, a planilha deve conter uma aba com as informações da estrutura e tempos de transferências em forma de matriz adjacente.

### 3.2.2 Linguagem e estrutura do algoritmo

Dentro da solução proposta a linguagem escolhida foi Python. Além do desenvolvedor possuir experiência prática, Python é uma linguagem de programação interpretada, interativa e orientada a objetos. Ela fornece estruturas de dados de alto nível, como listas e matrizes associativas (chamadas dicionários), digitação dinâmica e vinculação dinâmica, módulos, classes, exceções, gerenciamento automático de memória, etc. Ela tem uma sintaxe extremamente simples e elegante e ainda é uma linguagem de programação poderosa e de propósito geral.

Como muitas outras linguagens de *script*, ela é gratuita, mesmo para fins comerciais, e pode ser executada praticamente em qualquer computador moderno. Um programa Python é compilado automaticamente pelo interpretador no código de *byte* independente da plataforma que é então interpretado. É possível executar Python em sistemas operacionais como Linux, Windows NT, 98, 95, IRIX, SunOS, OSF [23].

Manter a estrutura lógica do algoritmo é um pré-requisito, salvo as alterações que resolvam os problemas descritos anteriormente.

### 3.2.3 *Business Intelligence*

No presente, a empresa possui licença da ferramenta Qlik Sense. Logo, por **motivos financeiros** e estreiteza dos analistas da empresa com a ferramenta foi definido que a aplicação deve ser desenvolvida utilizando Qlik Sense.

## 3.3 Extração de Dados

A extração de dados também pode ser dividida em duas fases: Extração do algoritmo e extração da ferramenta de *Business Intelligence*.

A tabela que contém os dados de entrada é uma tabela incremental semanal, ou seja, novos dados são inseridos na tabela a cada sete dias e não existe alteração nos dados do passado.

### 3.3.1 Extração de dados do algoritmo

Cada linha de produção armazena suas informações em um banco de dados diferente. Visto que todos os bancos estão alocados em servidores, os mesmos podem possuir diferentes *url*, com diferentes instâncias, nomes, *login*, senhas e nomes de tabelas. Para que o algoritmo consiga fazer a extração quando houver mudanças nos parâmetros ou quando houver a implementação em uma nova linha de produção, o algoritmo deve ler os parâmetros de conexão com o banco de dados do arquivo Excel. Além disso, existe a necessidade da utilização de um **drive** que faça a conexão do algoritmo em Python com o banco de dados.

Com os parâmetros de conexão corretos e o drive devidamente instalado, o **algoritmo** extrairá  $n$  linhas do banco de dados, onde  $n$  é um parâmetro que também pode ser inserido pelo usuário no arquivo Excel. Para uso do algoritmo, serão buscados os campos Identificação da ocorrência, Identificação da linha de produção, Identificação da estação, Início da ocorrência, Fim da ocorrência, Duração da ocorrência, Descrição da causa e *Flag* de tratamento. A extração de tais campos deve ocorrer com condicionais.

Apenas serão buscadas ocorrências que atenderem os requisitos:

- Linha de Produção

São necessárias apenas as linhas de produção descritas no arquivo Excel.

- Identificação da Estação

Devem ser tratadas apenas as ocorrências com estação igual a estação gargalo.

- Início da Ocorrência

A busca deve retornar apenas ocorrências com a data de início desejada inserida no arquivo Excel pelo usuário.

- Duração da Ocorrência

São necessárias apenas as ocorrências com duração maior do que o valor encontrado no arquivo Excel.

- Descrição da Causa

Devem ser tratadas apenas ocorrências com descrição das causas encontradas no arquivo Excel.

- *Flag* de Tratamento

A busca deve retornar apenas ocorrências com o campo *flag* de tratamento nulo.



### 3.3.2 Extração *Business Intelligence*

Por padrão o Qlik Sense utiliza um objeto próprio de conexão para fazer interface com os banco de dados. Muitos dos conectores que acessam essas fontes de dados estão integrados ao Qlik Sense, enquanto outros podem ser adicionados. Cada tipo de conexão de dados tem configurações específicas e caso o objeto não exista é necessária a criação [24]. Este processo é manual e para cada banco de dados diferente deve ser criada uma nova conexão.

Neste caso é necessária a criação da conexão do Qlik Sense com o banco de dados da empresa. Visto que cada linha de produção possui um banco de dados próprio, cada aplicação deve possuir sua própria conexão. Após as conexões estabelecidas, é possível iniciar o processo de extração. Este procedimento será melhor explicado na seção de ETL.

## 3.4 Implementações

Após definição das ferramentas de linguagem de programação - Python, interface de inserção de *inputs* - Excel e ferramenta de *Business Intelligence*, é necessária a implementação da conexão entre o algoritmo e a interface assim como o desenvolvimento da estrutura de ETL.

Para melhor interação entre o usuário e o arquivo Excel foram criadas duas abas, uma contendo as informações de parametrização e a outra a estrutura da linha e os tempos de transferência entre estações.

### 3.4.1 Melhorias Lógicas

Após revisão do algoritmo já implementado foi sugerido que o algoritmo não inserisse no banco de dados os resultados os quais não fariam diferença no BI, como por exemplo ocorrências que possuem porcentagens nulas. Tal mudança foi implementada com a simples verificação do campo de porcentagem resultante e em caso do valor ser igual a zero não há o processo de inclusão no banco.

### 3.4.2 Parametrização - Dados Gerais

A união entre o *script* e a interface se resume na implementação da configuração das variáveis de entrada do sistema. Assim o primeiro passo é a leitura do arquivo Excel utilizando a linguagem Python. Existem diversos módulos que desempenham esse papel como *xlrd*, *openpyxl*, *Pandas*, *xwlt*, *xlrd*, *csv* entre outras. Nesta solução foi escolhida a biblioteca *Pandas* por fornecer uma estrutura de dados rica e funções projetadas para fazer o trabalho com a estrutura de dados rápida, fácil e expressiva [25].

Utilizando *Pandas* é possível encontrar uma coluna da tabela pelo seu título e então ler os campos dessa coluna. Dessa forma, na aba de parâmetros, foi utilizada uma lista onde cada posição correspondia a uma coluna indicificada pelo título da tabela. Para a alocação dos dados da coluna da tabela na lista foram utilizados vetores, logo a estrutura de parâmetros se tornou uma lista de vetores contendo as mesmas informações da tabela de parâmetros.

Após as informações serem lidas e armazenadas na lista, variáveis globais recebem os dados da lista e as mesmas tornam-se disponíveis para o uso na lógica do algoritmo.

### 3.4.3 Parametrização - Estrutura e Tempos de Transferência

A lógica do algoritmo exige uma função que tenha como entrada o sentido do fluxo e o nome da estação referente, e que possua como saída a estação ou estações sequenciais e o tempo de propagação. A solução aplicada na parametrização da estrutura foi a utilização de uma matriz de adjacência (subseção 2.3). Conforme mencionado anteriormente, o arranjo da matriz de adjacência permite que seja definida a relação entre cada linha e coluna. Aqui, cada linha e coluna representa uma estação e todo valor diferente de zero remete a relação entre as estações. A Figura 7 abaixo representa um exemplo:

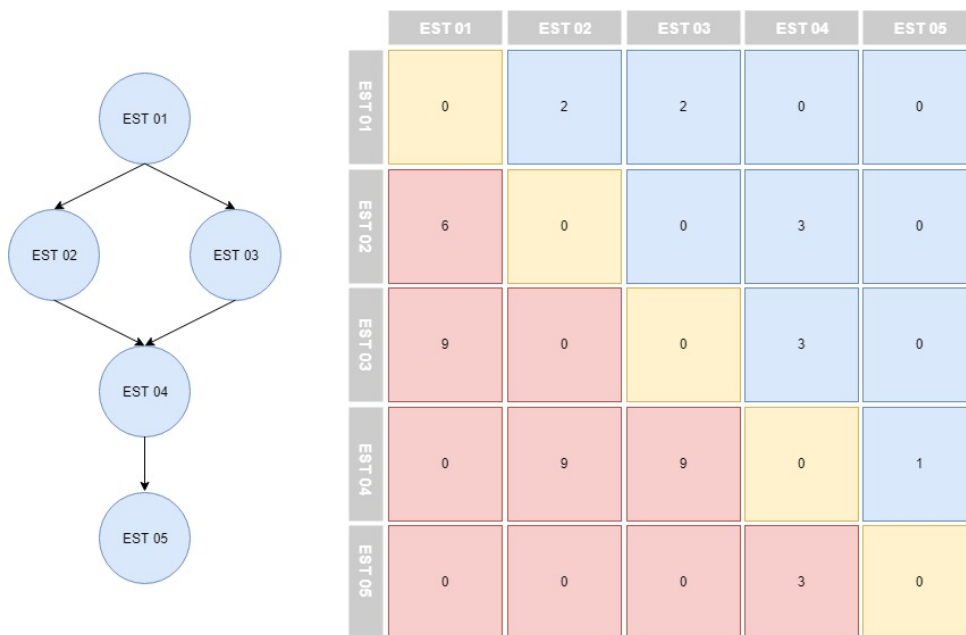


Figura 7 – Exemplo Matriz Adjacência e Grafo.

O valor zero indica que não há relação direta entre estações e o valor diferente de zero indica não somente que há relação, mas também o tempo de transferência entre as estações em segundos. Utilizando a matriz de adjacência, ainda é possível definir a direção do fluxo. Os valores acima da diagonal principal indicam o tempo de propagação de uma parada por bloqueio - sentido natural da linha de produção. Assim como os valores abaixo

da diagonal principal indicam o tempo de propagação de uma parada por falta de peça. Por exemplo, analisando a matriz é possível concluir que caso a estação 02 (dois) sofra qualquer parada da classe bloqueio, irá demorar três segundos para a estação 04 (quatro) entrar em bloqueio. Tal como caso haja uma parada da classe falta de peças na estação 03 (três) a estação 01 (um) entrará em modo de bloqueio após nove segundos.

Visto que essa estrutura se encontra no arquivo Excel, o algoritmo também utiliza a biblioteca *Pandas* para ler a matriz da seguinte forma: primeiramente é salva a primeira coluna contendo o nome das estações dentro de uma lista; então é definida uma nova lista que receberá apenas os valores da diagonal principal superior, onde cada índice da lista representa uma estação e cada posição da lista é um vetor contendo os tempos de transferência da estação referente. Da mesma maneira, é definida outra lista contendo apenas os valores da diagonal inferior. Logo, o algoritmo representa a estrutura da linha com três listas: a lista contendo os nomes das estações, a lista de tempos de transferência da classe bloqueio e lista de tempos de transferência da classe falta de peça.

Dada a exigência da lógica do algoritmo, foi implementada uma função que primeiramente procura dentro da lista de nomes de estações qual índice que o nome se encontra. Com a entrada da direção do fluxo, a função escolhe se olhará para a lista de bloqueio ou falta de peça e então com a lista e índice definidos, retorna os tempos requisitados assim como as estações que contém tais tempos.

### 3.4.4 Modelagem dos *Dashboards*

Sendo o foco principal do projeto auxiliar o controle das estações que causavam mais ocorrências e o tempo de cada ocorrência, foram escolhidas as informações essenciais que deveriam estar contidas no *Dashboard*, sendo elas:

- Quantidade de ocorrências

Esta é uma medida que refere a quantidade de ocorrência por estação.

- Duração de cada ocorrência

Esta é uma medida contendo o tempo total de duração de cada ocorrência agrupado por estação.

- Identificação das ocorrências

Esta é uma dimensão que conta com o número da identificação (ID) da ocorrência.

- Causas das ocorrências

Esta é uma dimensão que conta com a descrição da causa que gerou de cada ocorrência.



- Resultados das ocorrências

Esta é uma dimensão que conta com a descrição das causas que foram geradas devido as ocorrências.

- Estações Causadoras da ocorrência

Estações causadoras da ocorrência é uma dimensão que contém as estações que causaram ocorrências.

- Estação Resultante

Esta dimensão faz referência à estação gargalo.

- Data da Ocorrência

Esta é uma dimensão contendo a data da ocorrência.

Além de definir as informações que devem estar no *Dashboard*, é necessário também definir o *design* que será apresentado ao usuário final. Para isso, foram feitos alguns esboços com o objetivo de delimitar em qual parte da página ficaria cada gráfico, tabela, filtros e todas as informações que devem ser apresentadas.

### 3.4.5 Modelagem de Dados

Com o *design* do *Dashboard* definido, passou-se para a modelagem do banco de dados, no qual todas as informações serão armazenadas. Para tanto, foi utilizada apenas uma tabela Fato como mostrado na Figura 8.



FATO
Resulting - ID
Resulting - Equipment
Resulting - Date
Resulting - Duration in Seconds
Resulting - Reason
Causer - Reason
Causer - Equipment
Causer - Equipment/Reason

Figura 8 – Fato Resultante

### 3.4.6 ETL

Para alcançar a tabela Fato modelada anteriormente, foi necessário desenvolver um processo de extração(*extraction*), transformação (*transformation*) e carga (*load*). Para cada uma das três etapas foi desenvolvida uma aplicação dentro da ferramenta Qlik Sense.

- Extração

Como explicado no item 3.3.2, o Qlik Sense exige que seja estabelecida uma conexão entre a ferramenta e o banco de dados onde se encontra as tabelas a serem extraídas. Após o objeto de conexão ser devidamente configurado inicia-se o processo de carga de dados.

A carga de dados é realizada utilizando uma declaração **SQL Select** no banco onde se encontram as **tabelas input e output do algoritmo**. Foi escolhido carregar todos os campos das tabelas para caso haja necessidades futuras. Ao carregar as tabelas, as mesmas são salvas em arquivos nativos da ferramentas, os arquivos QVD. Um

arquivo QVD (QlikView Data) é um arquivo que contém uma tabela de dados exportada do Qlik Sense. Esse formato de arquivo é otimizado para velocidade na leitura de dados de um *script* e ao mesmo tempo é compacto. A leitura de dados de um arquivo QVD é geralmente de 10 a 100 vezes mais rápida do que a leitura de outras fontes de dados [26].

O arquivo que contém os dados da tabela *input* será chamado de *input.qvd* e o arquivo contendo os dados da tabela *output* será chamado de *output.qvd*. Com as duas tabelas salvas em seus devidos arquivos, inicia-se o processo de transformação.

- Transformação

Para simplificar o processo que será executado diariamente de forma automática, foi executado um processo que não será repetido. O processo pode ser descrito da seguinte forma: foi criado um arquivo QVD sem dados, mas com nome "*TransformationData.qvd*" onde *Data* segue o padrão "SemanaAno", ex: "Transformation012019.qvd", *Transformation* da semana um do ano 2019.

Em seguida, é carregado o arquivo *output.qvd* e então são carregados todos os dados de identificação (ID) de ocorrências tratadas contidos no arquivo. Depois são carregados os dados contidos na tabela *input* com a condição de apenas carregar os dados que contenham a mesma identificação de ocorrências carregadas anteriormente pela tabela *output*. Os campos carregados da tabela *input* são renomeados de acordo com a necessidade. Esses arquivos são salvos dentro do arquivo criado anteriormente "*TransformationData.qvd*".

Após execução do processo manual foi criado o processo que será executado diariamente de forma automática. O processo segue a seguinte lógica: são carregados todos os arquivos no formato "*TransformationData.qvd*" e os dados são concatenados em uma única tabela (para fins explicativos essa tabela é nomeada de "T1"). Então, dentro do campo data da ocorrência da tabela "T1" é pega-se a máxima data e salva dentro de uma variável "V1". Assim, é carregado apenas o campo de identificação de ocorrências do arquivo *output* e salvo dentro da tabela "T2". O próximo passo é carregar os dados do arquivo *input* onde a data da ocorrência seja maior que "V1" e a identificação de ocorrência exista na tabela "T2", após isso os dados são salvos na tabela "T3". A Figura 9 abaixo ilustra a operação.

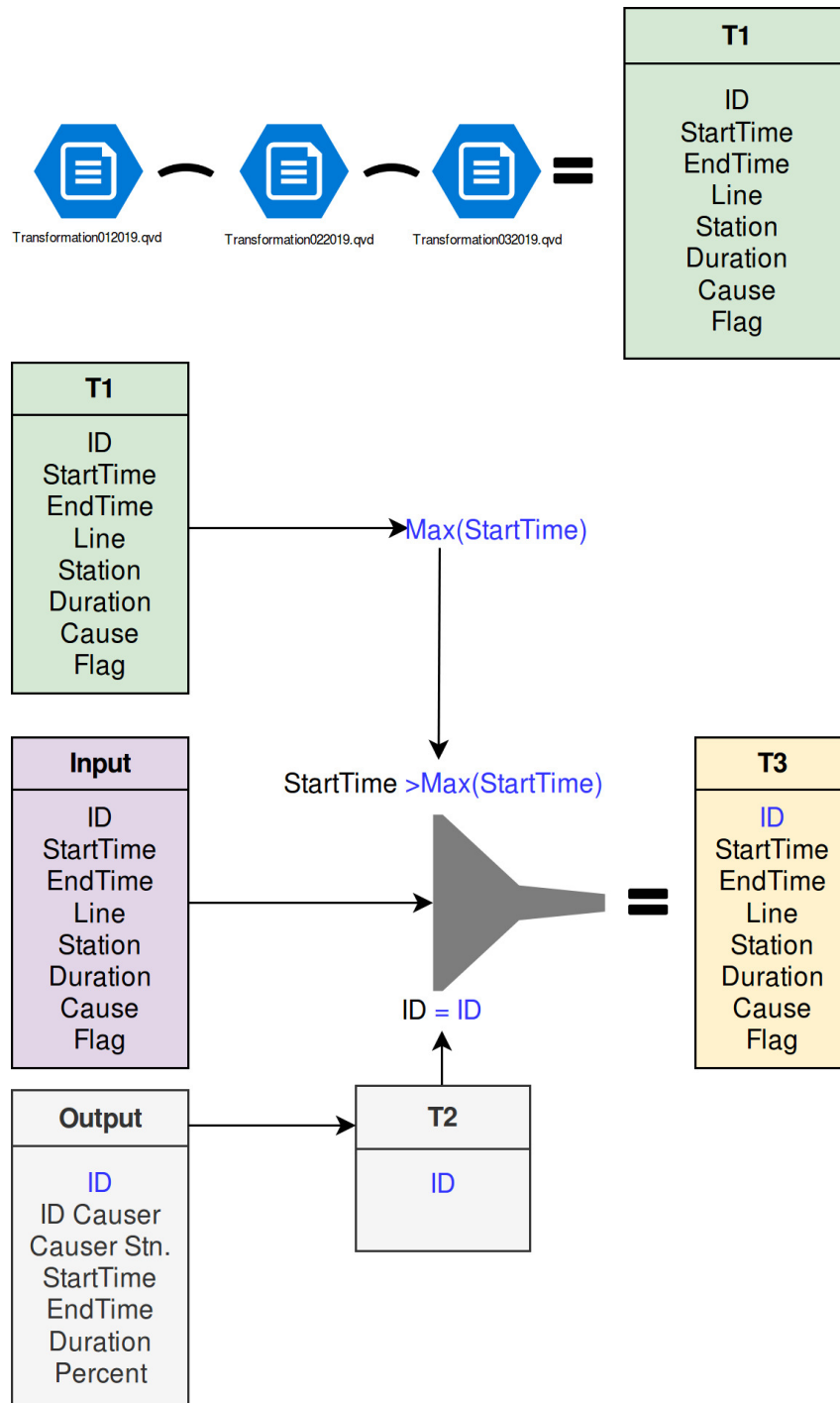


Figura 9 – Ilustração Transformação

O passo seguinte é checar se já existe o arquivo da semana atual, caso negativo o arquivo é criado e os dados da tabela "T3" são renomeados e salvos dentro do arquivo recém criado. Em caso positivo, os dados do arquivo da semana atual são carregados e os dados da tabela "T3" são concatenados dentro da tabela semanal, que é salva no arquivo existente como mostrado na Figura 10.

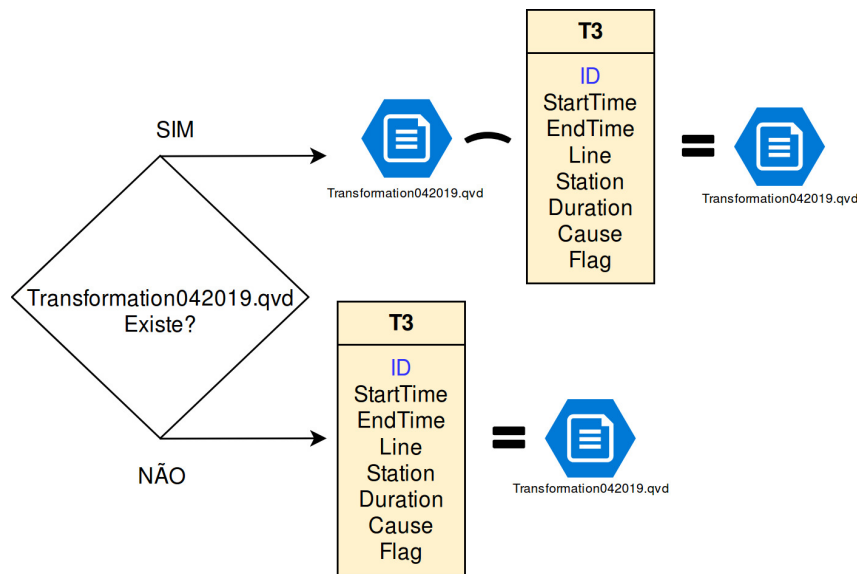


Figura 10 – Segundo Passo Transformação

Por fim, são carregados os dados do arquivo *output.qvd* e salvos dentro da tabela "T4". Então, são carregados todos os dados contidos nos arquivos de formato "*TransformationData.qvd*", esses dados são concatenados e salvos dentro da tabela "TF". Com a tabela "TF" devidamente carregada, é executado um *Left Join* com a tabela "T4", resultando em uma tabela "TF" com dados modelados do arquivo *input.qvd* unidos com os dados do arquivo *output.qvd*.

A tabela "TF" é salva em um arquivo *Fato.qvd* e se encerra a transformação, como ilustrado na Figura 11.

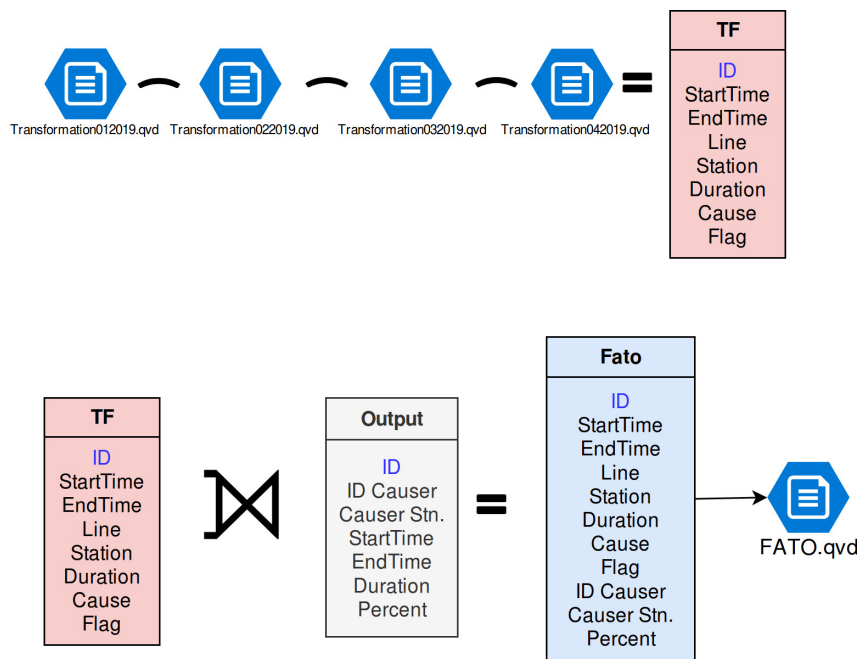


Figura 11 – Terceiro Passo Transformação



- Carga

A etapa de carga inicia-se com o carregamento do arquivo *Fato.qvd*. Os campos são renomeados de acordo com a necessidade e são criados os campos *Resulting - Duration in Seconds* e o campo *Causer - Equipment/Reason*. O primeiro deve ser calculado multiplicando o campo *Durataion* pelo campo *percent*. O segundo é o resultado da concatenação entre o campo *Equipment* e o campo *Cause*. Além disso, o campo que contém a data e hora do início da ocorrência é transformado em um campo que só contém a data.

O usuário exigiu que as estações deveriam possuir nomes específicos que ele definiu, então ficou acordado que esses nomes seriam escritos em tabelas de arquivos *.xlsx* pelo usuário com a identificação da estação ao lado do nome desejado. A fim de não haver erros na conexão com a tabela Fato, ficou definido que a descrição da estação gargalo estaria em uma tabela diferente das descrições das demais tabelas.

O último passo da etapa de *Load* é carregar as tabelas com as descrições das estações. Elas serão dimensões da tabela fato e se conectará com a mesma pelo campo Equipamento.

Com os campos devidamente calculados e as dimensões carregadas é formado o esquema estrela, que está pronto para ser usado na aplicação.

### 3.4.7 Criação do *Dashboard*

Com o processo de **ELT** bem desenvolvido e a tabela Fato a **dispor**, inicia-se a criação do *Dashboard*. O *Dashboard* é um painel que irá apresentar as informações mais relevantes para o usuário final, com o objetivo de auxiliá-lo nas tomadas de decisão.

Com o layout já definido, essa etapa consiste em criar o *layout* diretamente na aplicação. Para esse fim, o Qlik Sense possui uma interface chamada *Edit*, como mostra a Figura 12, que se define como serão dispostos os componentes no painel.

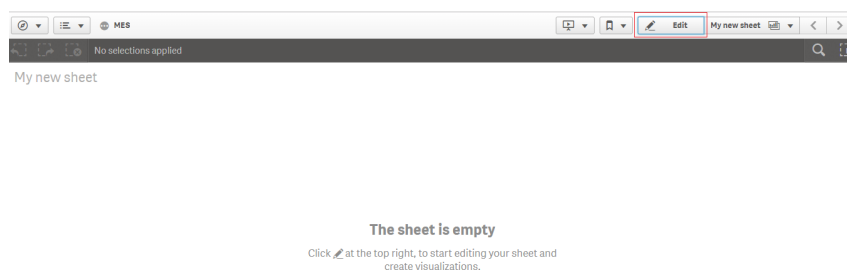


Figura 12 – Edição de Layout

Utilizando o esboço criado no início do projeto, foram definidos três *Dashboards* com as camadas de filtros, indicadores e gráficos.

- Filtros

Os filtros criados contém a dimensão de data, equipamento resultante, causa resultante e equipamento gerador. Todos os filtros foram criados utilizando o componente nativo *Filter* do Qlik Sense. Com a adesão do componente no painel foi necessário simplesmente fazer a inclusão das dimensões dentro do mesmo.

- Indicadores

Os indicadores inseridos foram de quantidade de falhas e tempo total de duração. Os dois indicadores criados foram inseridos no *Dashboard* utilizando o componente também nativo, *KPI*. Após a inserção, foi preciso apenas configurar os componentes para exibir as medidas.

- Gráficos

Utilizou-se gráficos de barra, gráficos de dispersão e *Sankey*. O gráfico de barras deve conter informação da dimensão *Causar - Equipment/Reason* e a medida da duração do tempo parado que cada equipamento e razão ocasionou. O gráfico de dispersão deve conter a dimensão de equipamento causador da ocorrência e a quantidade de ocorrências que o equipamento ocasionou. O gráfico *Sankey* deve conter as dimensões: equipamento resultante, causa da ocorrência e por fim equipamento gerador. A medida deve ser a duração total de cada parada em função da causa e o equipamento gerado.

Os gráficos de barras e dispersão são nativos, contudo o gráfico *Sankey* teve que ser adicionado utilizando uma extensão. Porém, todos os gráficos são configurados da mesma forma: primeiro o gráfico é adicionado no *Dashboard* e depois são configuradas suas dimensões e por fim suas medidas.

Todos os *Dashboards* possuem os mesmos filtros e indicadores variando apenas os gráficos. Dentro da ferramenta de edição do Qlik Sense, a função que possibilita a inserção dos componentes citados acima no painel é a função *Charts*, como mostra a Figura 13.

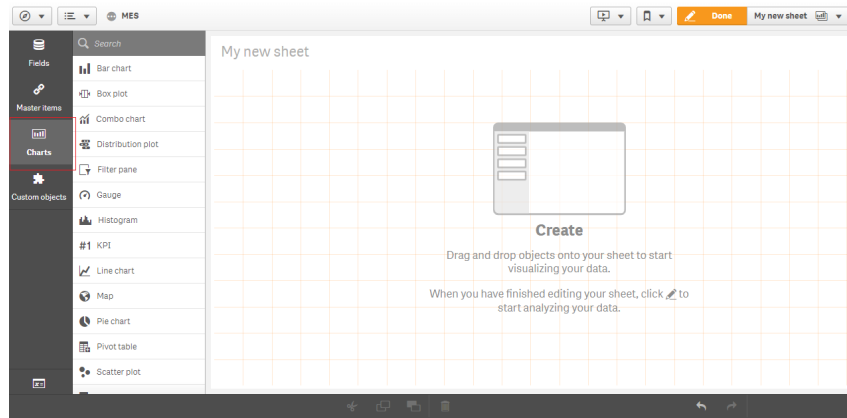


Figura 13 – Edição de Layout - Inserção de Componentes

### 3.4.8 Atualização Diária

O banco de dados da empresa é atualizado diariamente. Logo, as informações que são apresentadas no *Dashboard* desenvolvido também devem ser atualizadas todos os dias de forma automática.

Para o correto funcionamento do processo ETL é necessário que as atualizações ocorram na ordem correta, extração seguido da transformação e carga. A ferramenta que coordena esse sequenciamento dentro do *Qlik Sense* é o *Task* e ela se encontra dentro da área de administração da ferramenta chamada *QMC* [27]. A Figura 14 aponta a opção de configuração *Task*.

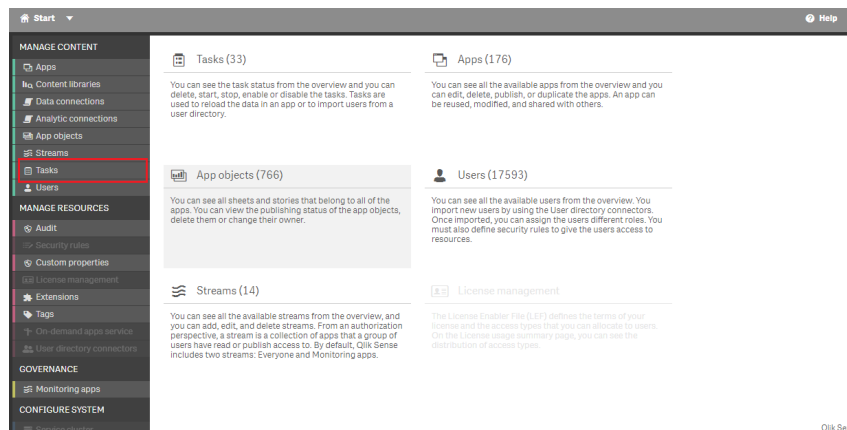


Figura 14 – Configuração Tasks

A configuração do *Tasks* inicia-se ajustando o gatilho da extração. Um horário fixo para essa extração iniciar foi definido baseado na troca de turno dos trabalhadores da linha. Dessa forma, o gatilho da extração será este horário definido. O próximo passo é a transformação, dado que o carregamento da aplicação depende do fim da extração, o gatilho da transformação foi definido com o término da extração. Da mesma maneira foi

configurado o início da atualização da aplicação da carga. O gatilho da carga é o fim da atualização da transformação.

Após a configuração da cadeia de atualização do Qlik Sense, o desenvolvimento do BI foi concluído.

## 4 Resultados

### 4.1 Algoritmo

No capítulo anterior foi visto que quanto ao algoritmo houve alteração na linguagem de programação para Python, parametrização via interface Excel e um ajuste lógico no algoritmo. Com todas essas alterações realizadas foram escolhidas aleatoriamente 100 ocorrências para teste e validação da alteração. Os critérios realizados para fim de comparação foram o de otimização, performance e generalização.

#### 4.1.1 Translação de Linguagem

A "tradução" do algoritmo da linguagem Java para Python não apresentou grandes obstáculos e um fator que contribuiu com o sucesso é da linguagem possuir grande volume de suporte disponíveis na internet, além de ser uma linguagem bastante utilizada na atualidade. Contudo, era uma premissa que a resposta dos dois algoritmos, utilizando os mesmos parâmetros de entrada, se mantivessem. Entretanto a resposta foi igual em noventa e quatro por cento das cem ocorrências testadas.

Ao investigar as seis ocorrências que não obtiveram o mesmo resultado, verificou-se que existe um parâmetro dentro do algoritmo antigo que quando duas ocorrências empatam em parâmetro específico, o resultado é aleatório na escolha da ocorrência que melhor atende as necessidades. Ou seja, dentro das cem ocorrências, ao rodar o algoritmo inúmeras vezes o mesmo pode apresentar resultados diferentes em no mínimo seis por cento das ocorrências. O mesmo teste foi feito no algoritmo desenvolvido em Python e o resultado se manteve em todas as interações.

Este evento não era de ciência da empresa. Os resultados citados acima foram apresentados ao usuário e assim, a translação de linguagem do algoritmo foi validada pelo mesmo.

#### 4.1.2 Performance

Quanto à velocidade de execução, foram testados dois casos, o primeiro com a função que não insere no banco resultados que não tem funcionalidade na aplicação de *Business Intelligence* e o segundo mantendo todos os resultados. A velocidade de execução do algoritmo em Java foi verificada pela ferramenta de IDE *NetBeans*. O algoritmo em Python foi testado utilizando a biblioteca nativa *Time*.

Quanto a versão que manteve todos os resultados, o algoritmo que teve melhor performance foi o algoritmo em linguagem Java, dois minutos e onze segundos, enquanto a



versão em Python teve o tempo de dois minutos e quarenta e um segundos.

Já no segundo teste, a versão em Python que ignorava as inserções de resultados desnecessários obteve melhor resultado de velocidade do que o algoritmo em Python sem alteração, mas ainda não foi mais rápido do que o algoritmo em Java. **Nota-se que essa diferença em relação ao algoritmo de mesma linguagem é devido ao tempo que os algoritmos gastam para fazer a comunicação com o banco de dados e inserir dados no mesmo.** A Tabela 2 apresenta a comparação performática dos algoritmos.

Linguagem	Tempo(s)
Java	131
Python (Sem alteração)	161
Python (Com alteração)	147

Tabela 2 – Comparação Performática

### 4.1.3 Generalização

A utilização do arquivo Excel como interface de parâmetros **agradou os usuários** e atendeu as necessidades do algoritmo. Foram feitos dois testes com duas bases distintas de diferentes linhas e houve integração completa de todos os parâmetros.

O arquivo Excel com parâmetros gerais possui o *layout* da Figura 15:

Server/Instance	DataBase	Login	Password	Start Data	Imput Table
url/xxx	DB	login	xxxxx	xx/xx/xxxx	input

Limit	Bottleneck	Duration[s]	Output_Table	Production_Line	Bloked_Causes	Starving_Causes	GF_Causes
100	174	00:00:05	output	1	Micro Parada	Micro Falta de Peça	Parada Genérica
					Bloqueio	Falta de Peça	

Figura 15 – Interface de Parametrização

**Quanto à parametrização do *layout* da linha utilizando a matriz de adjacência, o resultado foi satisfatório por atender as necessidades do algoritmo e agregar duas informações: sequência das estações e tempos de transferência. Contudo, notou-se que a complexidade para a criação do sequenciamento aumentou, sendo necessário um treinamento o usuário a parametrizar corretamente.** A Figura 16 exemplifica o modelo resultante da tabela de parametrização de *layout*.

Structure	St 01	St 02	St 03	St 07	St 50	St 70	St 98	St 117	St 139	St 174	St 190	St 210	St 215
St 01	0	8	0	0	0	0	0	0	0	0	0	0	0
St 02	28	0	8	0	0	0	0	0	0	0	0	0	0
St 03	0	16	0	7	6	0	0	0	0	0	0	0	0
St 07	0	0	24	0	0	11	0	0	0	0	0	0	0
St 50	0	0	21	0	0	12	0	0	0	0	0	0	0
St 70	0	0	0	18	33	0	14	0	0	0	0	0	0
St 98	0	0	0	0	0	15	0	13	5	0	0	0	0
St 117	0	0	0	0	0	0	14	0	0	1	0	0	0
St 139	0	0	0	0	0	0	39	0	0	3	0	0	0
St 174	0	0	0	0	0	0	0	15	9	0	12	0	0
St 190	0	0	0	0	0	0	0	0	0	29	0	4	0
St 210	0	0	0	0	0	0	0	0	0	0	17	0	9
St 215	0	0	0	0	0	0	0	0	0	0	0	28	0

Figura 16 – Interface de Parametrização de Layout

O principal resultado do algoritmo é referente a generalização. Como explicado anteriormente, o algoritmo só funcionava para uma linha de produção caso seus parâmetros internos não fossem alterados. Agora, o novo *script* é funcional para qualquer linha que for parametrizada na nova interface com o usuário como ilustrado na Figura 17 e na Figura 18.

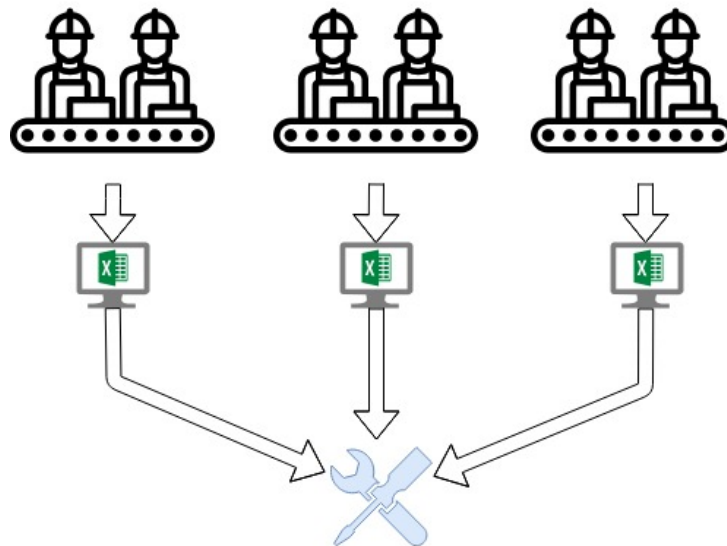


Figura 17 – Script Novo - N linhas de produção

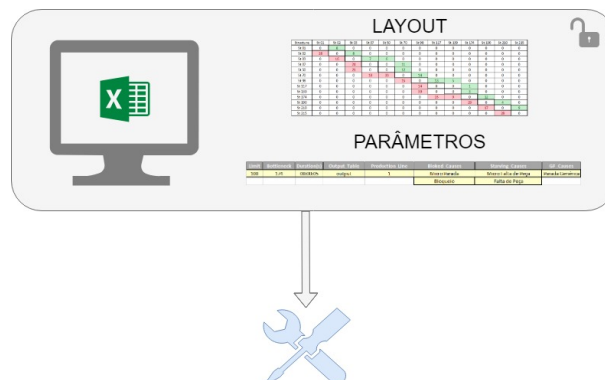


Figura 18 – Script Novo - Parâmetros variáveis

## 4.2 *Business Intelligence*

### 4.2.1 Extração

Com a conexão entre o Qlik Sense e o banco de dados parametrizada corretamente, o processo de extração foi executado com sucesso, alcançando seu objetivo: a armazenagem das tabelas de *input* e *output* salvas dentro dos arquivos QVD. O sucesso do processo de extração é mostrado na Figura 19



Figura 19 – Extração

### 4.2.2 Transformação

A modelagem da transformação realizada na seção 3.4.6 também alcançou seu objetivo. A tabela Fato estava devidamente modelada, os valores dos campos não sofreram nenhuma alteração inesperada, não houve duplicação de linha nas operações com tabelas e o processo foi executado com uma média de sete segundos como mostrado na Figura 20.



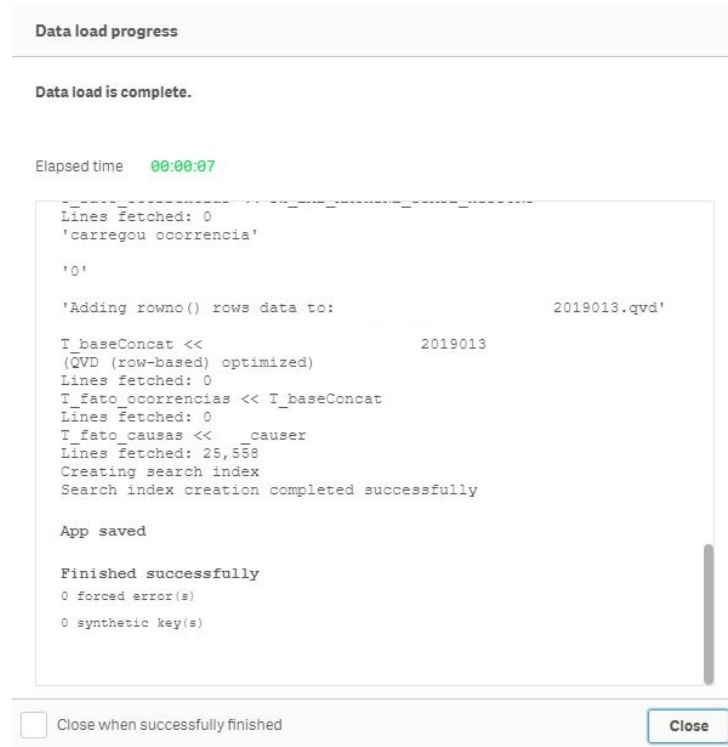


Figura 20 – Transformação

### 4.2.3 Carga

Também foi obtido êxito no processo de carga. Os campos calculados responderam como deveriam, o tempo de execução dessa etapa é em média de quatro segundos. Figura 21.

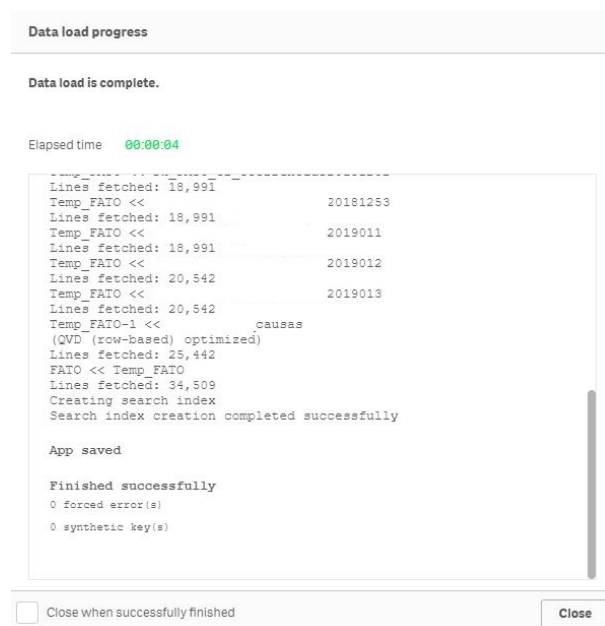


Figura 21 – Carga

A modelagem do esquema estrela também foi bem sucedido como mostrado na Fi-

gura 22.

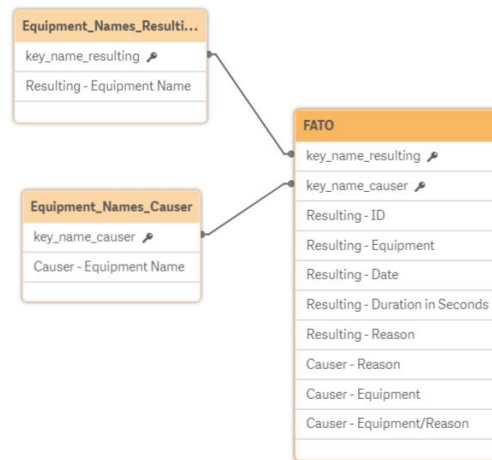


Figura 22 – Esquema Resultante

Quanto aos *Dashboards*, todos tiveram êxito em sua criação e aprovação do usuário. Todo o *layout* dos Dashboards foram projetados em contato com o usuário e durante o processo de criação houve alterações, como por exemplo o modelo dos filtros e indicadores.

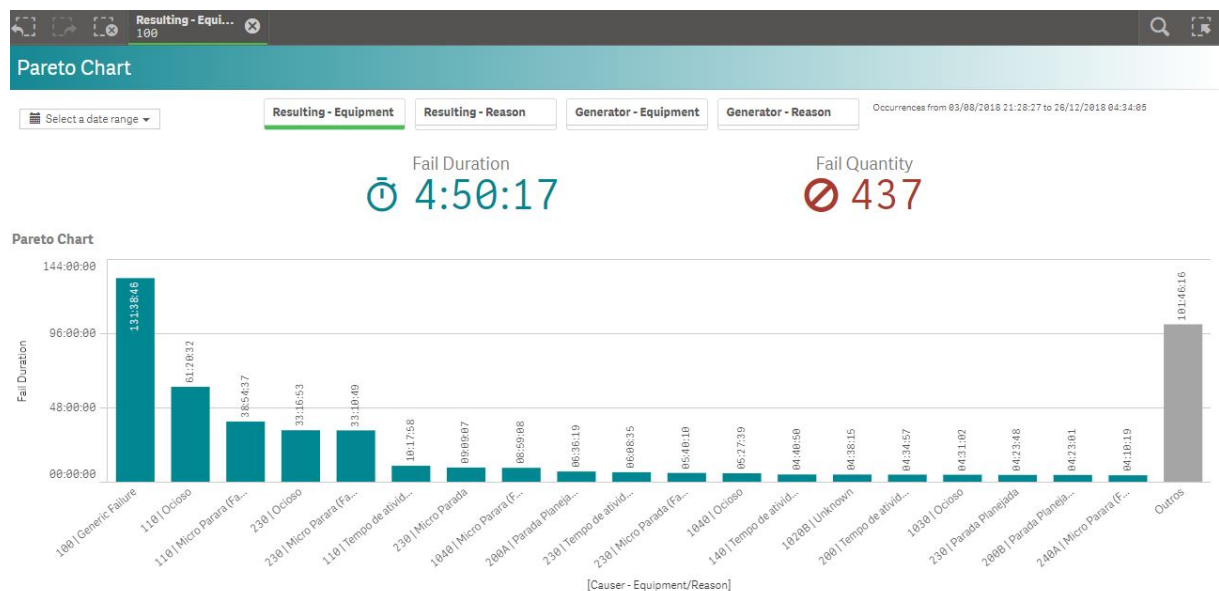


Figura 23 – Dashboard - Gráfico de Barras

O *Dashboard* apresentado na Figura 23 é o painel que mostra o gráfico de barras. Na primeira camada estão os filtros, a segunda camada os indicadores de duração total e quantidade de falhas e a última o gráfico propriamente dito. O primeiro filtro é o filtro de data, aqui optou-se por um filtro no qual é possível a seleção de intervalo de datas além de uma única data. O modelo do filtro é mostrado na Figura 24.

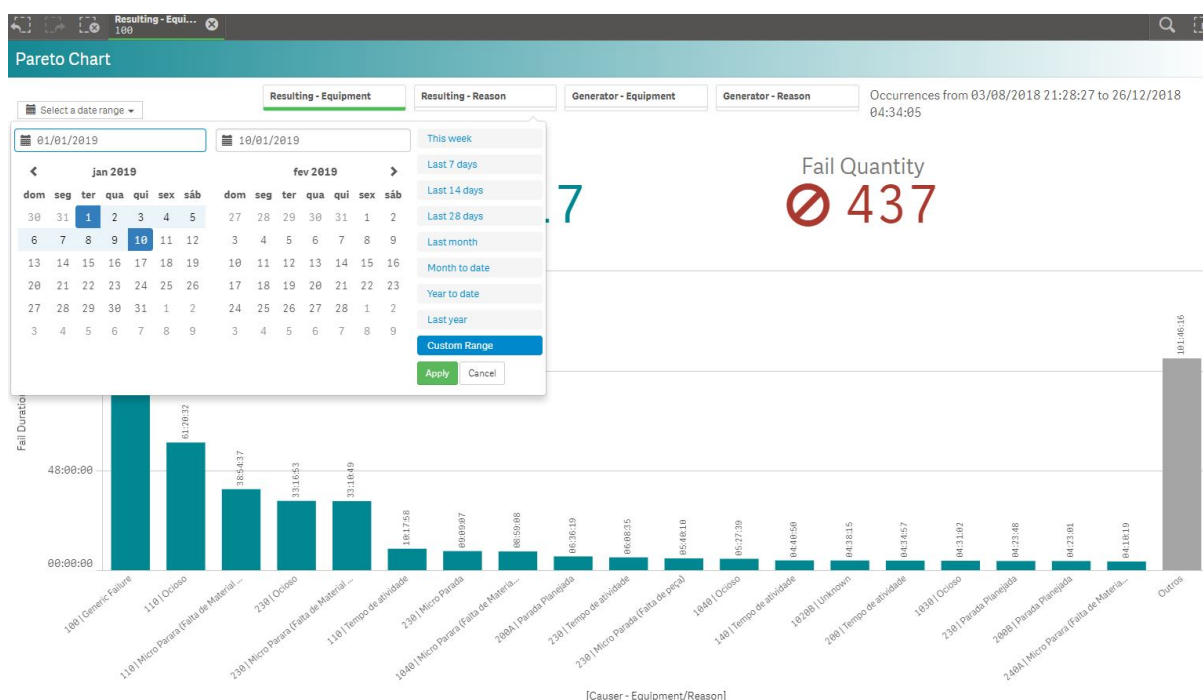


Figura 24 – Dashboard - Filtro de Data

O filtro *Resulting - Equipment* faz referência a estação gargalo. Hoje, há apenas uma estação gargalo parametrizada no sistema, contudo há a possibilidade de mudanças. Já o filtro *Resulting - Equipment* possibilita o usuário selecionar as causas das ocorrências na estação gargalo que foram pré definidas no algoritmo. O filtro *Generator - Equipment* proporciona ao usuário selecionar a estação que causou ocorrências. Por fim, o filtro *Generator - Reason* possibilita que o usuário selecione a causa que gerou a parada no gargalo. Nessa visão será possível mostrar todas as estações que causaram ocorrências devido a causa selecionada.

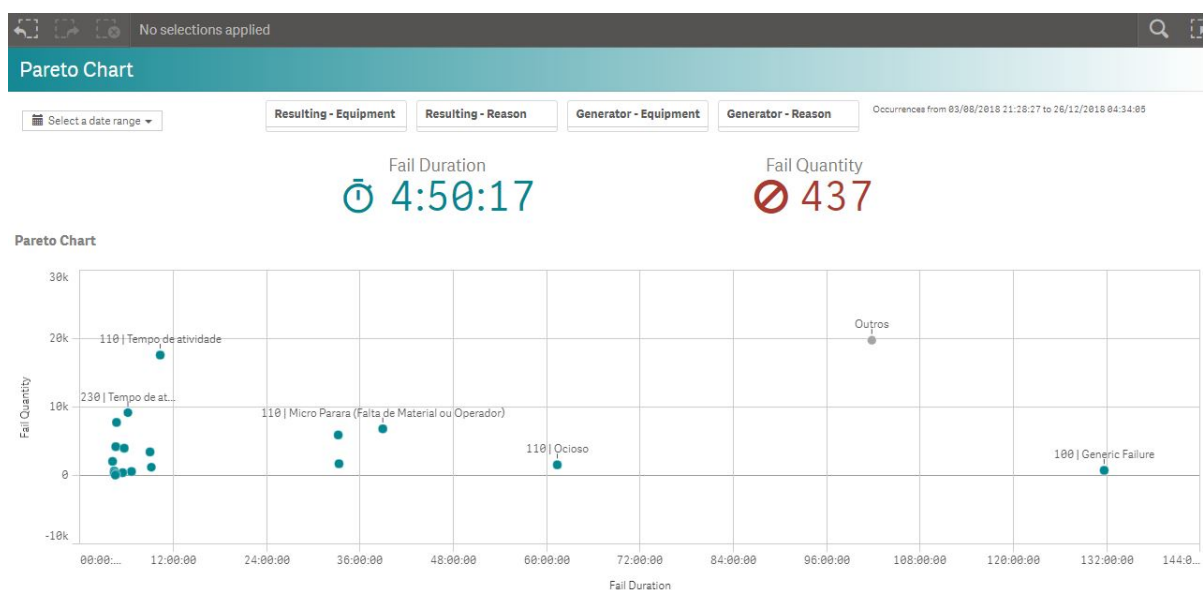


Figura 25 – Dashboard - Gráfico de Dispersão

O *Dashboard* do gráfico de dispersão mostrado na Figura 25 possui os mesmos filtros e indicadores criados no *Dashboard* do gráfico de barras. Esse gráfico foi criado com o seu eixo horizontal contendo a duração das ocorrências e o eixo vertical possui a medida de quantidade de ocorrências. Apesar do gráfico não possuir grande apelação visual, apresentou ser de grande utilidade para o usuário por possibilitar uma visão que os outros gráficos não proporcionam, dois eixos com medidas e nenhuma dimensão.



Figura 26 – *Dashboard* - Gráfico *SanKey*

Por fim, o *Dashboard* da Figura 26 mostra o gráfico *SanKey*. Com o mesmo padrão dos *Dashboards* anteriores, os filtros e indicadores possuem as mesmas informações e posições. Pode-se afirmar que o resultado foi satisfatório por trazer as informações corretas além de uma visualização única. O usuário da aplicação respondeu com *feedback* positivo ao visualizar este *Dashboard* e destacou o apelo visual que o gráfico possui.

#### 4.2.4 Atualização Diária

O processo de atualização diária também obteve um resultado satisfatório. O agendamento da extração ocorreu no horário pré-determinado, ativando o processo de transformação e por seguinte a atualização da carga. É possível notar na Figura 27, coluna *Status*, que todas as aplicações rodaram com sucesso.

Start

Tasks

Tasks Total: 35 Showing: 3 Selected: 1

Name	Associated resource	Type	Enabled	Status
EXT - MES	EXT - MES	Reload	Yes	Success
LOAD - MES	LOAD - MES	Reload	Yes	Success
TRA - MES	TRA - MES	Reload	Yes	Success

Edit

Delete

Start

Stop

Create new

More actions

Figura 27 – Agendamento ETL

## 5 Conclusões

O desenvolvimento desse projeto ocorreu na empresa BIX Tecnologia e foi obtido uma conclusão satisfatória: A ampliação da usabilidade de uma ferramenta já implementada e a implementação de um sistema de *Business Intelligence* para sistemas de manufatura e execução.

Os processos individuais do projeto foram validados, porém em conjunto ainda estão em fase de validação. Vale ressaltar que o sistema não foi completamente implementado em uma nova linha de produção. O novo algoritmo foi testado em uma nova linha, mas o sistema de BI foi implementado sobre uma linha já existente.

A troca de tecnologia do algoritmo foi bem aceita pelo usuário por ser considerada uma atualização do mesmo. Apesar do algoritmo antigo ter se apresentado mais **perfor-mático**, a nova linguagem é de conhecimento maior dos envolvidos que farão manutenção e ampliação do sistema.

No decorrer da mudança de tecnologia foi observado que poderiam ter sido feitas outras melhorias que podem aumentar a performance da ferramenta. Contudo, para respeitar a premissa de não alterar a lógica da ferramenta, não foram realizadas essas modificações.

A interface parametrizável pelo usuário também obteve aprovação do mesmo, visto que existem mudanças nos parâmetros gerais, como as instâncias do banco de dados, recorrentes na empresa. É válido salientar que com o objetivo do projeto alcançado, o usuário final possui mais autonomia na mudança de tais parâmetros. Assim, a empresa ganha com dispensabilidade de mão de obra qualificada para realizar alterações. Ainda sobre esse tópico, o algoritmo não apresentou uma resposta esperada quando os parâmetros foram inseridos incorretamente pelo usuário.

Apesar de no decorrer do curso Engenharia de Controle e Automação sejam vistas matérias de linguagem de programação e banco de dados, vários obstáculos foram encontrados no decorrer do desenvolvimento do projeto pelo motivo do curso não abordar temas ligados a *Business Intelligence*. Todavia, a empresa BIX Tecnologia ofereceu o suporte necessário para o desenvolvimento da ferramenta de BI.

Quanto às aplicações de *Business Intelligence*, todas atenderam ao seu propósito. Os conceitos teóricos de ETL e **modelo estrela** ampararam o desenvolvimento do projeto. A interação com o usuário durante o desenvolvimento dos *Dashboards* também foi crucial para a aceitação e satisfação do projeto. O usuário relatou que a aplicação possibilita as análises necessárias que antes não eram possíveis.

Atualmente o projeto inclui um algoritmo de avaliação de ocorrências e a apresentação de um sistema de *Business Intelligence* com as informações mais importantes para o usuário aplicadas a apenas uma linha de produção. Conquanto, após os desenvolvimentos



realizados nesse projeto, possui disponibilidade para expansão.

## 5.1 Trabalhos Futuros

Com os objetivos desse projeto alcançados, muitas oportunidades de melhoramento do sistema surgiram. Entre elas a conexão entre o término da execução do algoritmo com o início da extração do sistema de *Business Intelligence* e o aumento do número de estações gargalos a serem tratadas pelo algoritmo. Outro ponto, é que em um mesmo banco de dados, na mesma tabela, vão haver ocorrências de mais de uma linha, e um próximo passo seria o algoritmo tratar tal modificação. Também há a possibilidade de modificar o algoritmo para tratar linhas de produção que contenham mais de uma esteira. Outros pontos importantes seriam o tratamento de parâmetros inseridos incorretamente e melhorias na interface parametrizável pelo usuário.

Quanto ao sistema de BI além da ampliação no número de *Dashboards* e novos gráficos e da inclusão de novos dados ligados às estações encontradas em outras tabelas, também existe a possibilidade da migração da lógica do algoritmo para a linguagem nativa do Qlik Sense.

# Referências Bibliográficas

- 1 HOLST, L. Integrating discrete-event simulation into the manufacturing system development process. 2001. 13
- 2 UGARTE B., A. A. Saenz de; PELLERIN, R. Manufacturing execution system – a literature review. production planning control. 2009. 13
- 3 WATSON, H. J.; WIXOM. The current state of business intelligence. 2007. 13
- 4 FORTULAN EDUARDO VILA, G. F. M. R. Uma proposta de aplicação de business intelligence no chão-de-fábrica. *Gestão e Produção*, v. 12, n. 1, p. 55–66, 2005. 17
- 5 ALMEIDA M ISHIKAWA, J. R. M. S.; ROEBER, T. Getting started with datawarehouse and business intelligence. Agosto 1999. 17, 18, 19
- 6 PIEDADE, M. B. G. da. Business intelligence no suporte ao conceito e à prática de student relationship management em instituições de ensino superior. 2011. 17, 22
- 7 KIMBALL, R.; ROSS, M. In: *The Data Warehouse Toolkit Third Edition: The Definitive Guide to Dimensional Modeling*. [S.l.: s.n.], 2013. 17, 19, 20, 21, 22, 23, 24
- 8 NEGASH, S. Business intelligence, communications of the association for information systems. 2004. 17
- 9 TURBAN, E. e. a. Business intelligence: um enfoque gerencial para a inteligência do negócio. In: BOOKMAN EDITORA. [S.l.], 2009. 17
- 10 SANTOS, R. D. C. D. Power bi: A experiência de implantação em um escritório de contabilidade. 2018. 18, 20
- 11 HITACHI VENTARA. In: . [S.l.]. 20
- 12 MICROSOFT. Documentação do power bi. In: MICROSOFT. [S.l.]. 20
- 13 AB., Q. I. In: QLIK. [S.l.], 2011. 21
- 14 ZAPPAROLLI L., S. I. B. J. P. E. Aplicando técnicas de business intelligence e learning analytics em ambientes virtuais de aprendizagem. *VI Congresso Brasileiro de Informática na Educação (CBIE 2017)*, 2017. 21
- 15 ROWEN IY SONG, C. M. E. E. W. An analysis of many-to-many relationships between fact and dimension tables in dimensional modeling. 2001. 22, 23
- 16 GOLFARELLI DARIO MAIO, S. R. M. The dimensional fact model: A conceptual model for data warehouses. 1998. 22, 23
- 17 FERREIRA MIGUEL MIRANDA, A. A. e. J. M. J. O processo etl em sistemas data warehouse. 2010. 23
- 18 EL-SAPPAGH, A. M. A. H. A. H. E. B. S. H. A. A proposed model for data warehouse etl processes. 2010. 23, 24



- 19 OLSZAK, C. M.; ZIEMBA, E. Business intelligence systems in the holistic infrastructure development supporting decision-making in organisations. *Interdisciplinary Journal of Information, Knowledge and Management*, 2016. 23, 24
- 20 COTTENCEAU B., H. L.-B. J.-L.; FERRIER. Model reference control for timed event graphs in dioids. 2001. 24
- 21 PRESTES, E. In: UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL. *Introdução à Teoria dos Grafos*. [S.l.], 2016. 24, 25
- 22 JAN. Formal verification of java code generation from uml models. 27
- 23 SANNER, M. F. Python: A programming language for software integration and development. In: THE SCRIPPS RESEARCH INSTITUTE. [S.l.]. 30
- 24 AB., Q. I. Create a connection. In: QLIK. [S.l.]. 32
- 25 MCKINNEY, W. In: O'REILLY MEDIA, INC. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. [S.l.]. 32
- 26 AB., Q. I. Trabalhando com arquivos qvd. In: QLIK. [S.l.]. 37
- 27 AB., Q. I. Managing tasks and triggers. In: QLIK. [S.l.]. 42

# Apêndice

- **Extração**

[output]:

```
LOAD "Resulting - ID",  
"Causer - Equipment",  
"Causer - Code Reason",  
"Causer - Inicial Time",  
"Causer - Final Time",  
"Causer - Share",  
"Causer - Reason",  
"Causer - Equipment/Reason"  
FROM [lib://qvdfiles/MESoutput.qvd] (qvd);  
Store output into [lib://qvdfiles/output.qvd] (qvd);  
drop table output;
```

[input]:

```
LOAD  
"Resulting - ID",  
"Resulting - ProductionLine",  
"Resulting - Equipment",  
"Resulting - Start Time",  
"Resulting - Date",  
"Resulting - Day",  
"Resulting - Month",  
"Resulting - Year",  
"Resulting - Day/Month",  
"Resulting - Month/year",  
"Resulting - End Time",  
"Resulting - Duration",  
"Resulting - Duration in Seconds",
```

```
"Resulting - Reason",  
"Resulting - Causer"  
FROM [lib://qvdfiles/MESinput.qvd] (qvd);  
Store input into [lib://qvdfiles/input.qvd] (qvd);  
drop table input;
```

- **Transformação**

```
let vWeek = Week(today()-1);  
let vMonth = num( month(today() - 1), '00');  
let vYear = year(today()-1);  
T-tempData:  
LOAD  
[Resulting - Start Time]  
FROM [lib://qvdfiles/Transformation*.qvd] (qvd);  
NoConcatenate  
T-maxData:  
LOAD  
max([Resulting - Start Time]) as maxDateLoad  
RESIDENT T-tempData;  
DROP TABLE T-tempData;  
let vMaxDate = peek('maxDateLoad', 0, 'T-maxData');  
DROP TABLE T-maxData;  
trace 'Max Data already transformed: $(vMaxDate)';  
T-filtro-comcausa:  
LOAD  
[Resulting - ID] as ID  
FROM [lib://qvdfiles/output.qvd] (qvd);  
[T-fato-ocorrencias]:  
LOAD  
"Resulting - ID" ,  
[Resulting - ProductionLine],  
[Resulting - Equipment],
```

```

[Resulting - Start Time],
date( [Resulting - Start Time], 'DD/MM/YYYY') as [Resulting - Date],
day( [Resulting - Start Time]) as [Resulting - Day],
month( [Resulting - Start Time]) as [Resulting - Month],
year( [Resulting - Start Time]) as [Resulting - Year],
date(makedate(year( [Resulting - Start Time]), month( [Resulting - Start Time]), day(
[Resulting - Start Time]) ) , 'DD/MMM') as [Resulting - Day/Month],
date(makedate(year( [Resulting - Start Time]), month( [Resulting - Start Time]), day(
[Resulting - Start Time]) ) , 'MMM/YY') as [Resulting - Month/year],
[Resulting - End Time],
interval("Resulting - Duration", 'hh:mm:ss') as [Resulting - Duration],
num(Interval(Time("Resulting - Duration",'hh:mm:ss'),'ss')) as [Resulting - Duration in
Seconds], [Resulting - Reason],
[Resulting - Causer]
FROM [lib://qvdfiles/input.qvd] (qvd)
WHERE Exists("Resulting - ID") AND num([Resulting - Start Time]) > '$(vMaxDate)'
+ 0.00002;//adiciona 1 seg
DROP TABLES T-filtro-comcausa;
let vFileExists = isnull(QvdCreateTime('[lib://qvdfiles/Transformation$(vYear)$(vMonth)$(vWeek)
if (vFileExists + 1) then
trace 'Adding rowno() rows data to: sk-FATO-CD-ocorrencias$(vYear)$(vMonth)$(vWeek).qvd';
NoConcatenate
T-baseConcat:
Load
*
FROM [lib://qvdfiles/Transformation$(vYear)$(vMonth)$(vWeek).qvd] (qvd);
Concatenate(T-fato-ocorrencias)
LOAD *
RESIDENT T-baseConcat;
DROP TABLE T-baseConcat;
STORE T-fato-ocorrencias INTO [lib://qvdfiles/Transformation$(vYear)$(vMonth)$(vWeek).qvd]
(qvd);

```

```
DROP TABLE T-fato-ocorrencias;

else

trace 'Creating: FATO-CD-ocorrencias$(vYear)$(vMonth)$(vWeek).qvd';

STORE T-fato-ocorrencias INTO [lib://qvdfiles/Transformation$(vYear)$(vMonth)$(vWeek).qvd]
(qvd);

DROP TABLE T-fato-ocorrencias;

end if;

NoConcatenate

T-fato-causas:

LOAD distinct

[Resulting - ID],

[Causer - Equipment],

[Causer - Reason],

[Causer - Equipment/Reason],

[Causer - Inicial Time],

[Causer - Final Time],

[Causer - Share]

FROM [lib://qvdfiles/output.qvd] (qvd);

STORE T-fato-causas INTO [lib://qvdfiles/causer.qvd] (qvd);

drop table T-fato-causas;

FATO:

LOAD "Resulting - ID",

"Resulting - ProductionLine",

"Resulting - Equipment",

"Resulting - Start Time",

"Resulting - Date",

"Resulting - Day",

"Resulting - Month",

"Resulting - Year",

"Resulting - Day/Month",

"Resulting - Month/year",
```

```

"Resulting - End Time",
"Resulting - Duration",
"Resulting - Duration in Seconds",
"Resulting - Reason",
"Resulting - Causer"
FROM [lib://qvdfiles/Transformation*.qvd](qvd)
where "Resulting - Day" >= "$(vEndDate)" ;
Left Join (FATO)
LOAD
"Causer - Reason", "Resulting - ID", "Causer - Equipment", "Causer - Equipment/Reason", "Causer - Share" FROM [lib://qvdfiles/output.qvd] (qvd);
Store FATO into [lib://qvdfiles/Fato.qvd] (qvd);
Drop table FATO;

```

- **Carga**

FATO:

Load

```

"Resulting - ID",
"Resulting - ProductionLine" ,
"Resulting - Equipment",
"Resulting - Start Time",
"Resulting - Date",
"Resulting - Day",
"Resulting - Month",
"Resulting - Year",
"Resulting - Day/Month",
"Resulting - Month/year",
"Resulting - End Time",
"Resulting - Reason",
"Resulting - Causer",
"Resulting - Duration"*"Causer - Share" as "Resulting - Duration",
"Resulting - Duration in Seconds"*"Causer - Share" as "Resulting - Duration in Seconds",

```

```
"Causer - Reason",
"Causer - Equipment",
"Causer - Equipment" as key-causer,
"Causer - Equipment/Reason",
"Causer - Share"
from [lib://qvdfiles/Fato.qvd] (qvd);
Equipment-Names-Causer:
LAOD
Equipment-Code as key-causer,
Equipment-Desc as "Causer - Equipment Name"
from [lib://EXT-SQLServer/Equipment-Names.qvd] (qvd);
Equipment-Names-Resulting:
LOAD
Equipment-Code as key-resulting,
Equipment-Desc as "Resulting - Equipment Name"
from [lib://EXT-SQLServer/Equipment-Names.qvd] (qvd);
exit script;
```