





Universidade do Minho
Escola de Engenharia

Renata do Carmo Gonçalves
(A86440)

Benchmark de Tecnologias de Cloud Data Warehouse Existentes no Mercado Atual

Dissertação de Mestrado
Mestrado Integrado em Engenharia e Gestão de
Sistemas de Informação

Trabalho efetuado sob a orientação do
Professor Doutor Jorge Oliveira e Sá

DIREITOS DE AUTOR

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada. Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho



Atribuição

CC BY

<https://creativecommons.org/licenses/by/4.0/>

AGRADECIMENTOS

A realização deste trabalho não seria possível sem o apoio, disponibilidade e paciência de inúmeras pessoas que me acompanharam ao longo de todo o processo. A essas pessoas, manifesto aqui o meu sincero agradecimento.

Ao meu orientador, o Professor Jorge Oliveira e Sá agradeço pelo acompanhamento, pela ajuda prestada, e pela prontidão e confiança que demonstrou.

À minha família, principalmente aos meus pais e ao meu irmão deixo o meu agradecimento pelo constante apoio e motivação que me deram. Agradeço também ao meu grupo de amigos e ao meu namorado pela peça fundamental que foram ao longo de todo este percurso e pela motivação que me deram para continuar.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio, nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Benchmark de Tecnologias de Cloud Data Warehouse Existentes no Mercado Atual

Ao longo das últimas duas décadas a forma como os recursos de computação são desenvolvidos, implementados, atualizados e pagos foi drasticamente alterada, onde cada vez mais as soluções de software e hardware são transferidas para tecnologias *cloud*. Os Data Warehouses (DW), definidos como uma forma de organizar os dados corporativos de maneira integrada, num histórico variável no tempo e de modo a gerar uma única fonte de dados, também foram afetados com esta evolução, surgindo assim o conceito de Cloud Data Warehouse (CDW). Esta tecnologia permite que os utilizadores sejam mais livres tecnologicamente, pois não necessitam de despendar tempo a investir em software e hardware, pagam apenas pelos recursos utilizados e a infraestrutura em si apresenta uma maior flexibilidade e escalabilidade. No entanto, selecionar a tecnologia mais adequada para um CDW pode ser uma tarefa complexa, devido ao grande número de fatores que podem influenciar a decisão e devido à oferta existente no mercado, e as empresas devem estudar e entender cada plataforma antes da sua tomada de decisão final, de modo a selecionarem o que melhor corresponde aos seus requisitos. O objetivo da presente dissertação é a análise de um conjunto de plataformas de CDW presentes no mercado atual, com o objetivo de realizar um *benchmarking* entre estas, após a construção de um ambiente de CDW em cada uma, apurando assim as suas vantagens e desvantagens. Essas plataformas são, respetivamente, o Snowflake, o Google BigQuery, o Amazon Redshift, o Azure Synapse. Para tal, serão analisadas e investigadas características tais como arquitetura, escalabilidade, segurança e conformidade, suporte de dados e preços, e ainda medidas e avaliadas métricas como o tempo de carregamento de dados e de execução de *queries*. Todo o processo de construção dos CDW e avaliação do seu desempenho encontra-se descrito no presente documento.

Palavras chave: Armazenamento de Dados; Computação em Nuvem; Tecnologias de Armazenamento de Dados na Nuvem.

ABSTRACT

Benchmark of Cloud Data Warehouse Technologies Existing in the Current Market

Over the past two decades, the way computing resources are developed, deployed, upgraded, and paid for has changed dramatically, with more and more software and hardware solutions being transferred to cloud technologies. Data Warehouses (DW), defined as a way of organizing corporate data in an integrated manner, in a variable history over time and in order to generate a single data source, were also affected by this evolution, thus giving rise to the concept of Cloud Data Warehouse (CDW). This technology allows users to be more technologically free, as they do not need to spend time investing in software and hardware, they only pay for the resources used and the infrastructure itself has greater flexibility and scalability. However, selecting the most suitable technology for a CDW can be a complex task, due to the large number of factors that can influence the decision and due to the existing offer in the market, and companies must study and understand each platform before their final decision, in order to select the one that best matches their requirements. The objective of this dissertation is to analyze a set of CDW platforms present in the current market, with the objective of performing a benchmarking between them, after building a CDW environment in each one, thus determining their advantages and disadvantages. These platforms are, respectively, Snowflake, Google BigQuery, Amazon Redshift, and Azure Synapse. To this end, characteristics such as architecture, scalability, security and compliance, data support, and pricing will be analyzed and investigated, as well as metrics such as data loading and query execution times. The entire process of building CDWs and evaluating their performance is described in this document.

Keywords: Cloud Computing; Data Warehouse; Cloud Data Warehouse Technologies.

ÍNDICE

Direitos de autor	iv
Agradecimentos	v
Declaração de integridade	vi
Resumo.....	vii
Abstract	viii
Lista de abreviaturas/Siglas.....	xi
Lista de figuras.....	xii
Lista de tabelas.....	xiii
1. Introdução.....	1
1.1 Enquadramento e Motivação	1
1.2 Objetivos e Resultados Esperados	2
1.3 Abordagem de Investigação.....	3
1.4 Estrutura do Documento	5
2. Revisão de Literatura	6
2.1 Processo de Revisão de Literatura	6
2.2 Cloud Computing.....	8
2.3 Cloud Data Warehouse	11
2.4 Tecnologias de Cloud Data Warehousing	15
3. Seleção das Tecnologias de Cloud Data Warehouse	20
3.1 Introdução às Tecnologias Seleccionadas	20
3.2 Comparação das Tecnologias de Cloud Data Warehouse	23
4. Construção do Ambiente de Cloud Data Warehouse.....	32
4.1 Dataset: Star Schema Benchmark.....	32

4.2	Configuração dos Cloud Data Warehouses.....	41
4.3	Construção do Ambiente de Cloud Data Warehousing.....	41
4.4	Definição das Métricas a Testar.....	45
5.	Resultados Obtidos	48
6.	Conclusão.....	52
	Referências.....	54
	Apêndices	57
	Apêndice A – Criação das Tabelas na SA	57
	Apêndice B – Criação das Tabelas na SADL	59
	Apêndice C – Stored Procedures para Carregamento de Dados na SADL	61
	Anexos.....	76
	Anexo A – Estrutura das Tabelas do Schema TPC-H.....	76
	Anexo B – Construção da Base de Dados e da Staging Area	80
	Anexo C – Carregamento de Dados na Staging Area	84
	Anexo D – Interface do BQ Visualizer.....	87

LISTA DE ABREVIATURAS/SIGLAS

AWS	Amazon Web Services
BI	Business Intelligence
CC	Cloud Computing
CDW	Cloud Data Warehouse
CSV	Comma-separated Values
DDL	Data Definition Language
DLS	Data Lake Storage
DW	Data Warehouse
DWaaS	Data Warehouse as a Service
DWU	Data Warehouse Units
DSRM	Design Science Research Methodology
ERP	Enterprise Resource Planning
ETL	Extraction, Transformation and Load
GCP	Google Cloud Platform
GCS	Google Cloud Storage
IBM	International Business Machine Corporation
ML	Machine Learning
MPP	Massively Parallel Processing
MFA	Multi Factor Authentication
PaaS	Platform as a Service
RBAC	Role-based Access Control
SaaS	Software as a Service
SSB	Star Schema Benchmark
SQL	Structured Query Language
TPC-H	TPC Benchmark H
TI	Tecnologias da Informação
T-SQL	Transact-SQL
VPN	Virtual Private Networks

LISTA DE FIGURAS

Figura 1 - Design Science Research Methodology for Information Systems	3
Figura 2 - Processo Seguido na Elaboração da Revisão de Literatura	7
Figura 3 - Arquitetura de um DW	12
Figura 4 - Modelo Relacional TPC-H Benchmark.....	33
Figura 5 - Modelo Relacional SS Benchmark.....	34
Figura 6 - Criação da Base de Dados e SA no Snowflake	42
Figura 7 - DDL da Tabela CUSTOMER na SA	43
Figura 8 – Stored Procedure SADL_DIM_CUSTOMER_LOAD no schema SADL.....	44
Figura 9 - Medição do Tempo de Carregamento de dados para a tabela DIM_CUSTOMER... 46	
Figura 10 - Exemplo de Query do SSB	47
Figura 11 - Média dos tempos de carregamento dos dados nas respectivas tabelas da SADL. 48	
Figura 12 - Média dos tempos de execução das queries	50
Figura 13 - Criação de um novo Projeto no BigQuery.....	80
Figura 14 - Criação da SA no BigQuery.....	80
Figura 15 - Acesso ao Query Editor no Redshift.....	81
Figura 16 - Criação da Base de Dados e da SA no Redshift.....	81
Figura 17 - Criação da "Área de Trabalho da Synapse" no Azure Synapse	82
Figura 18 - Acesso ao Synapse Studio e criação de um Conjunto SQL dedicado.....	83
Figura 19 - Importação de Ficheiros CSV para o GCS.....	84
Figura 20 - Criação das Tabelas e Carregamento de dados para a SA no BigQuery	85
Figura 21 - Upload dos CSV para o DLS no Azure Synapse.....	86
Figura 22 - Criação das tabelas de SA e carregamento dos dados no Azure Synapse	86
Figura 23 - Interface do BQ Visualizer	87
Figura 24 - Tempo de Execução de Queries no BQ Visualizer	87

LISTA DE TABELAS

Tabela 1 - Escalabilidade nas Tecnologias de CDW.....	26
Tabela 2 - Segurança nas Tecnologias de CDW.....	27
Tabela 3 - Suporte de dados nas Tecnologias de CDW	28
Tabela 4 - Modelo de Preços das Tecnologias de CDW	31
Tabela 5 - Estrutura da Tabela FACT_LINEORDER.....	36
Tabela 6 - Estrutura da Tabela DIM_PART	37
Tabela 7 - Estrutura da Tabela DIM_SUPPLIER	38
Tabela 8 - Estrutura da Tabela DIM_CUSTOMER.....	38
Tabela 9 - Estrutura da Tabela DIM_DATE	39
Tabela 10 – Tamanho da Base de Dados TPC-H.....	40
Tabela 11 - Tamanho da Base de Dados SSB.....	40
Tabela 12 - Configuração das Tecnologias de CDW	41
Tabela 13 - Número de Linhas Retornado em cada Querie.....	47
Tabela 14 - Média dos tempos de carregamento dos dados nas respectivas tabelas da SADL	49
Tabela 15 - Média dos tempos de execução das queries	51
Tabela 16 - Estrutura da Tabela PART	76
Tabela 17 - Estrutura da Tabela SUPPLIER	76
Tabela 18 - Estrutura da Tabela PARTSUPP	77
Tabela 19 - Estrutura da Tabela CUSTOMER.....	77
Tabela 20 - Estrutura da Tabela NATION	77
Tabela 21 - Estrutura da Tabela ORDERS	78
Tabela 22 - Estrutura da Tabela LINEITEM.....	78
Tabela 23 - Estrutura da Tabela REGION.....	79

1. INTRODUÇÃO

Será inicialmente apresentado o enquadramento e motivação à realização da presente dissertação, bem como os objetivos e resultados esperados da mesma. De seguida, será descrita a abordagem de investigação selecionada para a elaboração da dissertação, e ainda desenvolvido um capítulo que descreve a estrutura do presente documento, de modo a oferecer um auxílio na leitura do mesmo.

1.1 Enquadramento e Motivação

Conforme nos movemos para uma sociedade orientada para a informação, a determinação de como organizar os dados para maximizar a sua utilidade torna-se um problema bastante relevante. Por essa razão, a gestão de dados tornou-se uma atividade de extrema importância nas organizações, uma vez que a correta manipulação de dados, além de dinamizar os diversos setores da organização, pode também fornecer indicadores que auxiliam a gestão estratégica das mesmas.

As empresas que mantêm um volume muito grande de dados espalhados em diversos sistemas optam por uma forma de obtenção de informações denominada de Data Warehouses (DW). Um DW organiza os dados corporativos de maneira integrada, com um histórico variável do tempo e uma única fonte de dados. Ao longo das últimas duas décadas a forma como os recursos de computação são inventados, desenvolvidos, implementados, atualizados, mantidos e pagos foi drasticamente alterada, onde cada vez mais as soluções de software e hardware são transferidas para tecnologia *cloud*. Ela possibilita que os indivíduos e as organizações possam obter acesso a um conjunto compartilhado de recursos ilimitados de TI, como servidores, armazenamento e aplicações, oferecendo uma ampla gama de vantagens aos seus utilizadores.

Desta evolução surgem os Cloud Data Warehouses (CDW) e podem ser vistos como uma infraestrutura física gerida por uma empresa fornecedora de serviço de *cloud*, o que significa que o cliente não precisa de fazer um investimento inicial em hardware ou software

e não precisa de gerir ou manter o DW. Assim, os utilizadores podem ser ainda mais livres tecnologicamente, focando-se na análise ou no uso dos seus dados.

Com a evolução dos sistemas *on-premise* para a *cloud*, também se sucedeu uma evolução na quantidade de fornecedores de serviços de CDW. Selecionar a tecnologia mais adequada para um CDW é uma tarefa bastante complexa, devido ao grande número de fatores que influenciam a decisão e devido à oferta existente no mercado atual. Embora as tecnologias de CDW compartilhem semelhanças em muitos aspetos, as principais têm diferenças significativas que as empresas devem estudar e entender antes da sua tomada de decisão final, sendo, portanto, um assunto que requer uma forte componente de investigação. Dependendo dos fatores mais relevantes para cada uma das organizações, o estudo de benchmarks de tecnologias de CDWs pode ser útil no sentido de estas terem uma base de conhecimento já previamente desenvolvida, acerca de um certo conjunto de métricas, facilitando assim o seu trabalho aquando da decisão de adoção de uma das tecnologias.

1.2 Objetivos e Resultados Esperados

A finalidade da presente dissertação é a realização de um *benchmark* de um conjunto de tecnologias de CDW presentes no mercado atual, através da construção de um ambiente de CDW em cada e da seleção de um conjunto de características e métricas a avaliar, com o objetivo de obter conclusões acerca do desempenho de cada uma. A seleção do tema da presente dissertação foi realizada em conjunto com a empresa na qual ela se encontra associada, a Deloitte Technology, e o orientador associado da Universidade do Minho.

Com a realização da presente dissertação pretende-se a obtenção dos seguintes resultados:

- Apresentação de uma revisão de literatura sobre a área de armazenamento de dados na *cloud*, discutindo os seus principais conceitos e estudando as gerações existentes deste tipo de tecnologias;
- Apresentação de um conjunto de tecnologias de CDW com maior presença no mercado atual e de um estudo de mercado de algumas previamente selecionadas,

relativamente a características como arquitetura, escalabilidade, segurança e conformidade, suporte de dados e preços;

- Construção de um CDW em quatro tecnologias de CDW, utilizando um *dataset* pré-definido; e
- Partilha dos resultados de um benchmark de quatro tecnologias de CDW, utilizando um conjunto de métricas, com o objetivo de obter resultados relativos ao desempenho de cada uma.

1.3 Abordagem de Investigação

Para o desenvolvimento da presente dissertação, foram seguidos os princípios, as práticas e os procedimentos estabelecidos pela Design Science Research Methodology (DSRM) for Information Systems (Peppers et al., 2007), de forma a gerir o processo de investigação e tornar cientificamente válidas as propostas e artefactos produzidos. Apesar da metodologia apresentar uma ordem sequencial, os autores admitem flexibilidade na sua aplicação, permitindo a iniciação de um projeto em qualquer um dos quatro “pontos de partida” representados na figura 1.

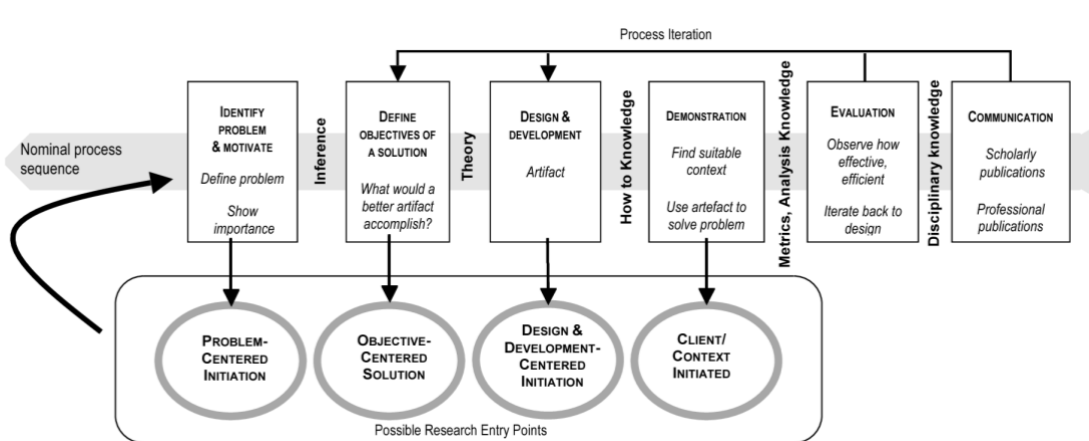


Figura 1 - Design Science Research Methodology for Information Systems

A metodologia DSRM inclui seis etapas, esquematizadas na figura 1, nomeadamente:

1. Identificação do Problema e Motivação: consiste na identificação dos aspetos que criam a oportunidade na definição do problema, que levarão ao desenvolvimento do

- projeto. É definida a situação atual, de forma a motivar e dar relevância ao trabalho desenvolvido, permitindo também esclarecer todo o processo;
2. Definição dos Objetivos: consiste na definição dos objetivos que permitem resolver o problema previamente identificado e obter o conhecimento necessário para pôr a solução em prática. Para tal, é necessário obter conhecimento dos métodos, tecnologias e teorias da área em questão;
 3. Conceção e Desenvolvimento: assim que se o problema e os objetivos estão definidos, é iniciada a revisão do estado de arte, sendo esta essencial para uma melhor compreensão dos principais conceitos associados à temática. Após o desenvolvimento destes capítulos, inicia-se a fase de conceção e desenvolvimento. Esta fase visa a evolução e progresso dos vários métodos e modelos necessários para a solução do problema. Nesta altura, uma proposta de arquitetura começa a ser desenhada, apesar de esta estar constantemente em mudança e a ser alinhada com o que é desejado para o projeto;
 4. Demonstração: consiste na validação do que foi previamente desenvolvido. Nesta fase, é provado se aquilo que foi desenvolvido é, ou não, capaz de ajudar na resolução do problema apresentado, envolvendo a definição de cenários de teste e a elaboração de testes;
 5. Avaliação: nesta fase são retiradas as conclusões do trabalho desenvolvido. É observado e avaliado em que medida é que a solução desenvolvida responde ao problema, comparando os objetivos com os resultados atingidos. Esta fase poderá levar a que tenham que ser feitos ajustes no que foi desenvolvido até ao momento, caso os resultados obtidos não sejam os desejados, ou se pretendam melhorias nos mesmos.
 6. Comunicação: esta fase visa apresentar e divulgar os resultados que foram obtidos no trabalho, realçando a oportunidade criada e a utilidade do trabalho desenvolvido. Esta fase inclui a escrita da dissertação, que se alonga ao longo de todo o trabalho, bem como outras oportunidades de divulgação do trabalho desenvolvido na comunidade científica (Geerts, 2011; Peffers et al., 2007).

1.4 Estrutura do Documento

O presente documento encontra-se organizado em cinco principais capítulos. No primeiro capítulo encontra-se identificado o enquadramento e a motivação para o desenvolvimento da presente dissertação, bem como são identificados os objetivos e resultados esperados. Também são apresentadas a abordagem de investigação adotada e a estrutura do presente documento.

O segundo capítulo apresenta a revisão de literatura dos conceitos base do tema desta dissertação. São explorados os conceitos de Cloud Computing (CC), DW e de CDW, incluindo as suas características, vantagens, entre outros temas essenciais ao seu entendimento básico. Por fim são ainda apresentados alguns exemplos de tecnologias de CDW, juntamente com capítulos explicativos do problema existente de seleção do fornecedor de CDW e de algumas características tipicamente analisadas entre as várias tecnologias.

No terceiro capítulo foi iniciado o desenvolvimento da dissertação, começando pelo estudo e descrição das quatro tecnologias selecionadas. Aqui é realizada uma comparação das quatro tecnologias tendo em conta um conjunto de características, respetivamente arquitetura, escalabilidade, segurança dos dados, suporte de dados e ainda modelos de preços.

No quarto capítulo foi inicializado o processo de construção dos CDW em todas as tecnologias. Neste capítulo podem ser encontradas informações acerca do *dataset* selecionado, incluindo o seu modelo relacional, descrição das tabelas e o tamanho dos dados, bem como informações acerca das configurações de cada uma das tecnologias, de forma a realizar o *benchmark* no máximo de condições similares. Adicionalmente são explicados os passos necessários para a construção destes ambientes, e, ainda, especificadas as definições das métricas que serão avaliadas no capítulo posterior.

O quinto capítulo apresenta os resultados e conclusões obtidas da medição das métricas mencionadas no capítulo anterior, alcançando assim o principal objetivo e resultado esperado da presente dissertação.

2. REVISÃO DE LITERATURA

No presente capítulo encontra-se apresentada a revisão de literatura efetuada sobre os conceitos base da temática CDW, inerente a esta dissertação. Inicialmente são investigados e apresentados os conceitos associados à CC, campo emergente da ciência da computação que encaminha o setor de Tecnologias da Informação (TI) para um novo nível. De seguida, são estudados, de forma aprofundada, vários tópicos acerca dos DW e dos CDW, apresentando assim as suas principais características, vantagens, bem como outros tópicos essenciais ao entendimento destes conceitos e da presente dissertação. Por fim, é realizada uma revisão de literatura acerca das tecnologias de CDW selecionadas para a concretização da presente dissertação, com o intuito de obter um conhecimento base sobre as mesmas.

2.1 Processo de Revisão de Literatura

A criação de um processo para a recolha da literatura relevante para a fase de enquadramento conceptual é essencial para, não só facilitar o trabalho de pesquisa e leitura, como para manter um método de trabalho consistente. Assim sendo, para a elaboração da revisão de literatura presente neste documento o principal foco foi a investigação dos principais conceitos relacionados à temática da presente dissertação. Os principais aspetos para a pesquisa e recolha de documentos foram a definição dos critérios de pesquisa, fontes e palavras chave utilizadas.

Para a pesquisa de documentos foram utilizadas algumas bases de dados de referência, nomeadamente Scopus, Google Scholar, Springer, RepositóriUM e em algumas situações o próprio motor de pesquisa da Google, nas quais foram utilizadas um conjunto de palavras chave de pesquisa. As palavras chave foram selecionadas a partir do tema da dissertação, sendo exemplos “data warehouse”, “cloud computing”, “cloud data warehouse”, “tecnologias de cloud data warehousing” entre outras relevantes. Na maioria das pesquisas foram aplicados filtros de datas entre 2010 a 2022, de modo a obter os mais recentes acontecimentos na área. No entanto, em determinados casos, através do estudo das referências bibliográficas de cada um dos documentos, foram utilizados artigos com anos inferiores, pois apresentavam informação histórica relevante à pesquisa. Adicionalmente, era

também aplicado o filtro de documentos que eram do gênero “Revisão de Leitura”. A recolha de documentos decorreu desde o início da elaboração da pré-dissertação, em novembro de 2021, até ao seu final, em fevereiro de 2022.

De modo a filtrar os artigos encontrados *online*, o critério mais utilizado foi primeiramente a leitura do título e palavras chaves do artigo em questão, seguida da leitura do seu resumo. Caso estes se encontrem dentro do tema pretendido, era realizada a leitura integral do documento. Por fim, eram analisadas as referências bibliográficas presentes em cada artigo, de modo a encontrar mais conteúdos possivelmente interessantes relacionados com a mesma temática. Relativamente à informação que não se encontrava em forma de artigo, como por exemplo *websites* e livros, o processo era bastante semelhante, lendo primeiramente o título e as primeiras páginas e avaliando a sua relevância para a presente dissertação. Através desta análise foi possível classificar os documentos como literatura relevante ou não relevante, selecionando assim os documentos mais interessantes. Os documentos ou *websites* obtidos deste processo foram lidos e referenciados na dissertação, quando o conteúdo se revelava adequado, sendo que numa fase inicial foram abertos cerca de 300 documentos e *websites*, lidos os resumos ou páginas iniciais de cerca de 150 e acabados por ser selecionados e referenciados cerca de 70. O processo descrito encontra-se ilustrado na figura 2.

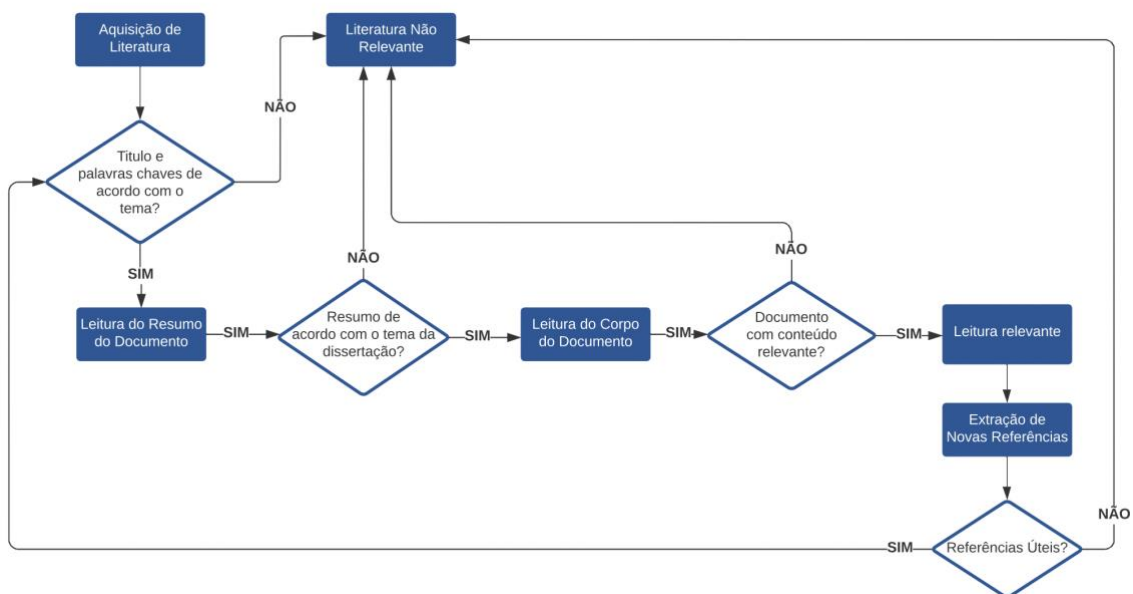


Figura 2 - Processo Seguido na Elaboração da Revisão de Literatura

2.2 Cloud Computing

No passado, as pessoas executavam as aplicações ou programas de software num computador físico ou num servidor perto de si. No entanto, ao longo das últimas duas décadas a forma como os recursos de computação são utilizados foi drasticamente alterada, onde cada vez mais as soluções de software e hardware são transferidas para tecnologia *cloud*. Ao longo do capítulo, o conceito de CC irá ser aprofundando, inicializando-se pela sua origem e conceito e pela comparação dos sistemas *cloud* aos sistemas *on-premise*. De seguida, serão abordadas as vantagens relacionadas com a adoção destes sistemas e ainda estudados os modos de implementação da *cloud*.

2.2.1 A Origem e Conceito da Cloud Computing

Encontramo-nos na era das tecnologias de CC, um campo emergente da ciência da computação que encaminha o setor de TI para um novo nível. A CC é uma evolução da TI e um modelo de negócios dominante para o fornecimento de recursos de TI, o que possibilita que os indivíduos e as organizações possam obter acesso de rede a um conjunto compartilhado de recursos ilimitados de TI, como servidores, armazenamento e aplicações (Sunyaev, 2020). A CC adotou o conceito de pagamento conforme o uso, onde os utilizadores dos serviços não precisam de pagar pela infraestrutura, nem pela sua instalação e manutenção. Qualquer pessoa pode aceder ao serviço desejado da *cloud* a qualquer hora, em qualquer lugar do mundo, sendo que os fornecedores de serviços em *cloud* oferecem cada vez mais um maior número e variedade de serviços, que podem escalar elasticamente de modo a atender às crescentes procuras de computação existentes (Haris & Khan, 2018). A CC não é um novo paradigma, mas sim um aprimoramento de várias tecnologias, como o hardware, a computação distribuída e as tecnologias baseadas na Internet. Ela fornece a infraestrutura que impulsionou as principais tendências digitais, incluindo a computação móvel, a Internet of Things, Big Data e a Inteligência Artificial, acelerando assim a dinâmica do setor, interrompendo os modelos de negócios existentes e fomentando a transformação digital (Sunyaev 2020).

2.2.2 Sistemas Cloud vs. On-Premises

Nas últimas duas décadas foram realizadas várias pesquisas acerca da definição e evolução da CC, visto que, devido ao seu rápido crescimento, esta foi definida como algo essencial em vários aspetos da sociedade, desde a educação, a instituições governamentais e a indústrias (Gill et al., 2019). Por exemplo, as empresas que desenvolvem os sistemas Enterprise Resource Planning (ERP) como a SAP, Oracle, Sage e Microsoft começaram a oferecer os seus ERPs também num ambiente baseado em *cloud*, o que não implica apenas uma mudança na utilização de recursos de computação para os clientes, mas também uma mudança profunda na lógica de criação de valor dos fornecedores e modelos de negócios dos seus parceiros. Com a introdução dos serviços por meio de CC, a forma tradicional de fornecimento de software aos clientes finais sofreu alterações, visto que nestas situações não existe nada para revender, nem para instalar tecnicamente e não há mais oportunidades de fornecer qualquer tipo de logística. A entrega de serviços em *cloud* é claramente diferente da entrega de sistemas de TI que acontecia até ao seu surgimento, o que implica uma transição de uma lógica dominante de bens para uma lógica dominante de serviço (Nieuwenhuis, Ehrenhard & Prause, 2018).

Assim, a CC atraiu muitas organizações para migrar os seus sistemas para a *cloud*, ou seja, para realizarem o processo de transição de todos ou parte dos seus recursos de TI, incluindo hardware, software, dados armazenados e processos de negócios, a partir de implementações *on-premise* para ambientes de *cloud*, onde podem ser geridos remotamente por terceiros. A decisão de migração acontece principalmente devido às suas variadas vantagens, particularmente a redução das despesas de capital e a capacidade de recursos virtualmente infinita, entre muitas outras que aumentam a agilidade das empresas e que irão ser descritas no capítulo seguinte. No entanto, esta é uma decisão organizacional estratégica que pode ser complicada, e que, portanto, requer a consideração e avaliação de uma ampla gama de aspetos técnicos e organizacionais (Alkhalil, Sahandi, & John, 2017).

2.2.3 Vantagens da Cloud Computing

A Internet e os serviços que se baseiam em CC dominaram as últimas duas décadas no mundo da TI, pois, com a emergente dependência do ser humano na tecnologia, a procura de

acesso aos recursos de computação, armazenamento e processamento dos dados tornou-se cada vez necessária. A CC apresenta-se como a solução para estas procuras, fornecendo instalações de computação disponíveis 365 dias por ano e 24 horas por dia em todo o mundo, através da Internet.

A CC tem sido alvo de bastante atenção por parte dos académicos e profissionais do sector, pois há cada vez mais serviços disponíveis na *cloud* para a realização de tarefas diárias orientadas para os utilizadores individuais, como o armazenamento de dados (Dropbox, OneDrive), a elaboração de documentos (Office 365, Google Docs), a gestão de negócios (SAP ByDesign) e os jogos online (GamingAnywhere, Stadia) (Sunyaev 2020). A CC apresenta também impactos nas organizações, pois possibilita aumentos de receitas e ajuda-as a atingir os seus objetivos de negócios, bem como outras características que motivam as organizações a migrar da infraestrutura local para *cloud*, nomeadamente:

- **Custo reduzido:** os recursos só são adquiridos quando necessários e pagos somente quando são utilizados. Adicionalmente, a infraestrutura não é comprada e, portanto, as despesas iniciais e os custos de manutenção são reduzidos;
- **Escalabilidade ilimitada:** é o principal benefício da tecnologia *cloud*, pois permite flexibilidade de aumentar ou diminuir a escala de acordo com as necessidades da organização. Assim, as organizações não precisam de se preocupar com as procuras futuras, pois podem adquirir facilmente os serviços adicionais a qualquer momento;
- **Flexibilidade:** as organizações são livres de decidir quais são os serviços que precisam e pagam de acordo com tal decisão, bem como estes serviços *cloud* podem responder de forma mais adequada às procuras de negócios, que se encontram em constante mudança;
- **Melhor mobilidade:** o acesso aos serviços da *cloud* podem ser efetuados a qualquer hora e em qualquer lugar, a partir de uma variedade de dispositivos, sendo que basta que estes estejam conectados à internet;
- **Upgrades mais fáceis:** é responsabilidade dos fornecedores de serviços *cloud* de atualizar a infraestrutura e os serviços para os seus clientes. As novas tendências e soluções de negócios são disponibilizadas rapidamente, permitindo às organizações competirem no mercado, adotando as tecnologias mais recentes;

- Recuperação de falhas: as organizações não precisam de estruturar planos complexos de recuperação de desastres ou falhas, ficando essa responsabilidade do lado dos fornecedores de serviços *cloud*;
- Segurança: pode ser um fator decisivo na escolha de um fornecedor de serviços *cloud*. Cada vez se dá mais importância às questões de segurança e privacidade dos dados que ficam armazenados na *cloud*. Dessa forma os fornecedores de serviços *cloud* investem avultadas quantias nos seus serviços e infraestruturas, de modo a oferecer melhor segurança (Ahmad, 2018).

No entanto, há organizações que ainda se encontram apreensivas em realizar a migração dos seus sistemas e serviços de TI para a *cloud*, devido à preocupação de perderem o controlo sobre eles ou de obterem outros resultados indesejados. Outras organizações estão interessadas em mover apenas alguns dos seus sistemas para a *cloud* devido à dificuldade de migrar as aplicações relacionadas com eles, por exemplo, software crítico para a segurança (Alkhalil et al., 2017).

2.3 Cloud Data Warehouse

À medida que as tecnologias se desenvolveram, muitas empresas promoveram o desenvolvimento de novos programas ou a transformação dos antigos para torná-los disponíveis por meio da *cloud*, incluindo os DW. Estas tecnologias oferecem bastantes benefícios, por um lado suportando as novas mudanças nos dados e, por outro lado, satisfazendo os novos desejos corporativos. Ao longo do capítulo, irão ser explorados estes conceitos, com especial foco nos CDW, abordando as suas vantagens e realizando uma comparação aos DWs *on-premises*, apurando assim as vantagens de migração para um sistemas na *cloud*.

2.3.1 Introdução aos Data Warehouses

De acordo com Moody e Kortink (2000), o conceito de DW consiste em organizar os dados corporativos de maneira integrada, num histórico variável no tempo, de forma integrada e não volátil. Ser orientado ao assunto significa que o DW permite efetuar análises

e facilita o processo de tomada de decisões para áreas específicas do negócio, o que torna a compreensão e a análise dos dados concisa e direta. Outra característica relevante dos é a perspectiva histórica que este mantém, ou seja, os DWs armazenam dados históricos, permitindo que o processo de tomada de decisão seja baseado no passado, de modo a perceber o presente e o futuro. Adicionalmente, os dados oriundos de diferentes fontes, passam por um processo de transformação e são armazenados no DW, fazendo com que o DW passe a ser a fonte única de dados que alimenta o processo de tomada de decisão podendo, portanto, ser classificado como um sistema integrado. Por fim, é ainda importante referir que as únicas operações possíveis de serem realizadas nos dados armazenados num DW é o carregamento inicial, posterior atualização e acesso de leitura, sendo que esta característica permite caraterizar o DW de não volátil (Singhal, 2021). Essa estrutura possibilita à empresa identificar tendências, de modo a posicionar-se estrategicamente no mercado, tornando-se mais competitiva e, conseqüentemente, aumentando os seus lucros (Raslan & Calazans, 2014).

Existe um conjunto de elementos básicos que constituem as arquiteturas dos DWs, sendo que esses podem ser as bases de dados, as ferramentas de Extract, Transform e Load (ETL), os metadados, os Data Marts, e ainda as ferramentas de acesso ao DW e a camada de relatórios, tal como é demonstrado na figura 3.

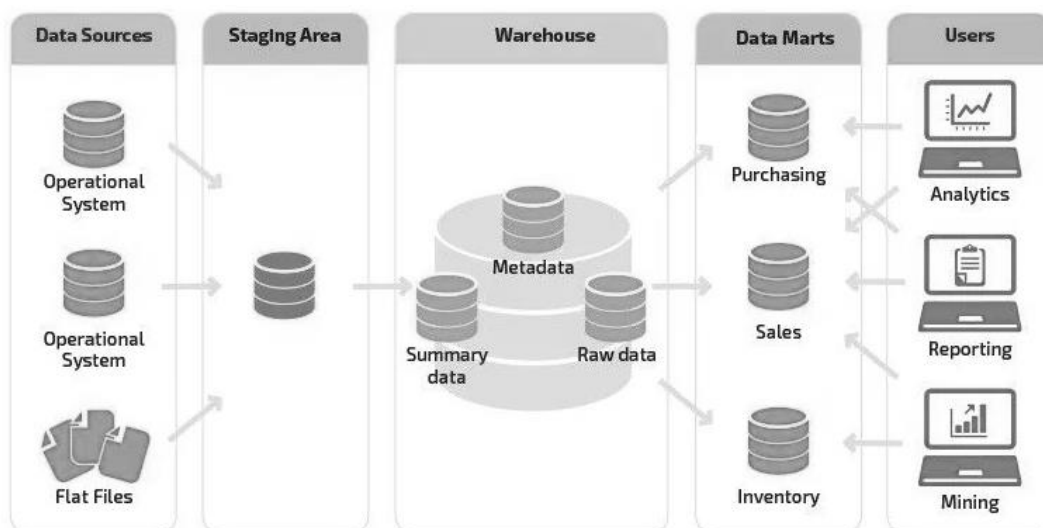


Figura 3 - Arquitetura de um DW

Nesta área de estudos, existe uma famosa metáfora apresentada por Kimball em 2002. Kimball afirma que um DW pode ser comparado a um restaurante, e num restaurante há duas zonas distintas, mas conectadas, a zona de preparação da comida, a cozinha, e a zona do serviço ao cliente, a sala do restaurante. Na zona da cozinha os chefes cozinham os pratos utilizando os melhores ingredientes disponíveis, inovando ao criar receitas através do uso de novas técnicas ou ingredientes. Na zona de serviço, o chefe de sala serve os pratos aos clientes, garantindo todo um ambiente adequado, de forma a melhorar a experiência do cliente. Num sistema de DW é também possível ver duas componentes: o processo de ETL que extrai os dados das fontes e que é composto pela limpeza, transformação e armazenamento dos dados no DW, e a segunda componente que consiste no acesso aos dados armazenados no DW, havendo o possível armazenamento dos dados em Data Marts orientados para um assunto, ou o acesso por parte dos clientes aos dados através de relatórios, visualizações gráficas, dashboards, etc. Tal como no restaurante, deve ser dada mais importância à experiência do utilizador no acesso aos dados que propriamente à construção do DW (Kimball & Ross, 2002).

2.3.2 Conceito e Características dos Cloud Data Warehouses

Recentemente, novos desenvolvimentos em tecnologias, como a criação da *cloud*, tornaram possível operar com novas quantidades e características de dados e as empresas que se baseavam em DW *on-premises* rapidamente se aperceberam que o sistema não seria capaz de acompanhar essa mudança. À medida que essas tecnologias se desenvolveram, muitas empresas diferentes promoveram o desenvolvimento de novos programas ou a transformação dos antigos para torná-los disponíveis por meio da *cloud* (Virant, R., Kamišalić, A., & Šestak, M., 2021).

A International Business Machine Corporation (IBM, 2021) define um CDW como uma infraestrutura física gerida por uma empresa fornecedora de serviço de *cloud*, o que significa que o cliente não precisa de fazer um investimento inicial em hardware ou software e não precisa de gerir ou manter o CDW. Ela permite que os utilizadores sejam ainda mais livres tecnologicamente, pois não necessitam de despende tempo e energia para investir em software e hardware, focando-se na análise ou no uso dos dados, de acordo com o que desejam produzir. A flexibilidade no contexto do CDW significa que a própria tecnologia pode

ser facilmente ajustada de acordo com os recursos necessários, o número de utilizadores e a localização geográfica (Rehman, Ahmad, & Mahmood, 2018). Outros atributos que suportam a funcionalidade e arquitetura dos CDWs são, nomeadamente:

- Elasticidade: o que significa que o sistema de CDW pode alocar e realocar dinamicamente as fontes de computação, incluindo armazenamento, processamento e fontes de rede. Ele aloca as necessidades e ajusta-as de acordo com os requisitos mínimos, utilizando apenas o que necessitam;
- Escalabilidade: vivendo numa época em que as perspectivas de dados produzem periodicamente quantidades incertas de dados, o CDW deve-se conseguir ajustar a esses requisitos. A escalabilidade sublinha a capacidade de aumentar ou diminuir instantaneamente, sem ter que se esperar ou migrar para outro servidor;
- Confiabilidade e disponibilidade: a confiabilidade refere-se à solidez do sistema, ou seja, em caso de erro, seja humano ou não, os dados ainda estão disponíveis para o utilizador final, o que também é frequentemente referido como recuperação de desastres;
- Eco-sistema: muitos fornecedores de CDW oferecem não apenas o próprio CDW, mas também outros componentes importantes, o que resulta num sistema coeso como um todo, que permite aos utilizadores operar dentro do sistema; e
- Multilocação: permitindo que vários utilizadores acessem, usem e operem no CDW (Virant, Kamišalić, & Šestak, 2021).

2.3.3 Vantagens de Migração para um Cloud Data Warehouse

O processamento de grandes quantidades de dados no DW requer enorme capacidade de processamento e espaço. Esses recursos são essenciais para que as empresas executem as suas operações diárias, e os departamentos de TI não podem fornecer poder de processamento adicional para o DW a qualquer momento. Nesta situação, a CC pode ser usada com sucesso para fornecer escalabilidade para períodos de pico, assim que for necessário, e apenas pagando pelo que foi usado. Esta possibilidade apareceu devido à competição existente entre os vários fornecedores de *cloud*, pois, ao haver competição, o desempenho da CC foi sendo aprimorado, começando a existir a hipótese de hospedar o DW

na *cloud*. A opção de migrar o DW para a *cloud* é também bastante executada devido à sua arquitetura de computador baseada em *cloud*, onde se pode aceder aos dados através de múltiplas fontes, estando estas na *cloud* ou num servidor remoto que é acedido através da internet (Thompson & Van der Walt, 2010).

Um DW tradicional oferece suporte total de Structured Query Language (SQL) e requer bastante tempo para configurar, otimizar e gerir o sistema, sendo uma das razões pela qual hoje em dia as organizações estão a migrar para CDWs. Assim, os clientes podem obter os dados de diferentes fusos horários e geográficos, ou seja, independentemente do local. Adicionalmente, existem inúmeros outros benefícios ao implementar um CDW. Estes, entre outros, incluem a rápida velocidade de implementação, o facto de oferecer serviços completos em tempo reduzido, de trazer confiabilidade e ainda um custo reduzido. Em termos de eficiência de custos, a infraestrutura ou hardware de hospedagem são caros, visto que o DW requer muitos softwares diferentes, portanto, os clientes podem economizar custos enormes utilizando a *cloud* e pagando pelo recurso de acordo com seu uso. Para as pequenas e médias empresas, também não é necessário se preocuparem com a propriedade e manutenção do software e hardware, o que traz um grande benefício para essas empresas. A CC permite então que as organizações gastem menos ou diminuam as suas despesas de capital, sendo apenas necessária uma despesa operacional que é muito baixa em comparação com as despesas de capital. Estas são algumas das inúmeras razões para a necessidade de migração dos serviços da organização para a *cloud* (Virant et al., 2021).

No entanto, dada a natureza proprietária e confidencial dos dados presentes num DW, a adoção da CC enfrenta vários desafios de segurança (Guermazi et al., 2015). Esses desafios, embora possam ser verificados minuciosamente, podem representar problemas de segurança, computação e rede, sendo que esses são frequentemente encontrados principalmente devido à natureza incompatível dos requisitos funcionais necessários para implementar o DW no ambiente de *cloud* e vice-versa (Awoyelu, Omodunbi, & Udo, 2013).

2.4 Tecnologias de Cloud Data Warehousing

Embora as tecnologias de CDW compartilhem semelhanças em muitos aspetos, apresentam também muitas diferenças significativas que as empresas devem estudar e

entender antes da sua tomada de decisão final. No presente capítulo, será abordada esta temática, que se apresenta como a motivação principal para o desenvolvimento da presente dissertação, bem como apresentadas algumas das razões influenciadoras aquando da escolha de um fornecedor de serviços *cloud*. Por fim, serão estudadas algumas das mais utilizadas tecnologias de CDW no mercado atual, de acordo com o *website* selecionado.

2.4.1 Seleção do Fornecedor de Serviços Cloud

Quando o tópico é escolher o fornecedor de CDW, as opções disponíveis no mercado são várias e cada fornecedor tem os seus benefícios e desvantagens. Portanto, as empresas necessitam de identificar o CDW que corresponde aos seus requisitos específicos. Esse é um aspeto importante da seleção e operacionalização dos serviços de migração na *cloud*. As organizações podem tentar migrar um pequeno conjunto de dados para vários CDWs, de modo a determinarem quais opções funcionam melhor para elas, tanto em termos de desempenho, como em termos de custos. Adicionalmente, também vale a pena notar que, ao migrar aplicações e cargas de trabalho para a *cloud*, os ambientes específicos que se escolher e os serviços oferecidos pelo fornecedor de serviços em *cloud* determinarão as configurações necessárias, o trabalho que vai ser necessário fazer e a ajuda que se vai poder obter por parte do fornecedor. Portanto, idealmente, deve-se selecionar os fornecedores de serviços *cloud* depois de identificar os candidatos à migração, mas fazendo isto paralelamente à análise e preparação das cargas de trabalho para a migração.

À medida que avançamos em 2022, a Nucleus antecipa que os clientes irão dar prioridade à respetiva escalabilidade de uma solução, aos recursos sem servidor, à abrangência multi-região e multi-nuvem e otimização de desempenho para *big data* e tarefas paralelas, ao selecionar uma solução de CDW. Portanto, ao longo do ano passado, os fornecedores concentraram o desenvolvimento em cinco áreas principais para diferenciar suas ofertas de CDW: provisionamento e atribuição de *clusters* eficientes, aproveitando algoritmos de Machine Learning (ML), recursos totalmente geridos e sem servidor, de modo a reduzir a supervisão administrativa e os custos associados, a abrangência da multi-nuvem e da multi-região de modo a maximizar a flexibilidade do cliente e reduzir o risco de interrupções específicas da *cloud* e da região, e ofertas de serviços para simplificar as implementações e mitigar os riscos de implementações mal configuradas (Wurm, 2022).

2.4.2 Características Tipicamente Avaliadas nos Cloud Data Warehouses

Como já foi previamente mencionado, quando se trata de selecionar um fornecedor de serviços *cloud*, existem requisitos e os critérios de avaliação específicos para a organização ou cliente em específico. No entanto, existem algumas áreas comuns de foco durante qualquer avaliação do fornecedor de serviços, nomeadamente:

1. **Arquitetura e serviços:** é necessário que o cliente se certifique que a plataforma do fornecedor e as tecnologias utilizadas estão alinhadas com o seu ambiente atual e apoiem os seus objetivos de *cloud*, bem como as arquiteturas, padrões e serviços de *cloud* do fornecedor devem atender às suas cargas de trabalho e preferências de gestão;
2. **Gestão e Segurança dos dados:** o local onde os dados residem e as leis locais subsequentes às quais estão sujeitos podem ser uma parte fundamental deste processo de seleção de fornecedores. Por exemplo, se o cliente tiver requisitos e obrigações específicas, este deve procurar fornecedores que lhe ofereçam escolha e controlo sobre a jurisdição em que os seus dados são armazenados, processados e geridos. O cliente deve ainda avaliar os níveis de dados e segurança do sistema, a maturidade das operações de segurança e os processos de governança de segurança;
3. **Custos:** embora nunca deva ser o fator único ou mais importante, não há como negar que o custo desempenhará um grande papel na decisão de qual fornecedor de serviços de *cloud* se deve escolher, visto que muitas empresas procuram migrar para a *cloud* também para reduzir os seus custos de produção. É importante que as empresas entendam como os custos serão medidos e calculados para vários serviços, para que possam planejar e definir melhor as suas necessidades;
4. **Escalabilidade:** uma das principais características dos serviços na *cloud* é a capacidade de escalar recursos. Portanto, a elasticidade oferecida pelos fornecedores de soluções de CDW deve ser bem estudada pois esta dá às organizações a liberdade de aumentar ou diminuir a escala conforme e quando considerarem necessário;
5. **Confiabilidade e Recuperação de Desastres:** deve-se procurar entender os processos de recuperação de desastres do fornecedor e sua capacidade de dar suporte às expectativas de preservação de dados do cliente. Isso deve incluir a criticidade dos

dados, fontes de dados, agendamento, *backup*, restauração, verificações de integridade, etc.;

6. Certificações e padrões: os fornecedores que cumprem com padrões reconhecidos e estruturas de qualidade demonstram uma adesão às melhores práticas e padrões do setor. Embora os padrões possam não determinar qual o fornecedor de serviços que se deve escolher, eles podem ser muito úteis na seleção de fornecedores em potencial;
7. Dependências e parcerias do serviço: Os fornecedores de serviços podem ter vários relacionamentos com outros fornecedores, e avaliar este relacionamento, os seus níveis de credenciais, capacidades técnicas e certificações é algo que pode influenciar à decisão; e
8. Saúde do negócio e perfil da empresa: avaliar as capacidades técnicas e operacionais de um fornecedor em potencial é obviamente importante, mas deve-se também considerar a sua saúde financeira e o seu perfil (Cloud Industry Forum, 2020).

2.4.3 Exemplos de Tecnologias de Cloud Data Warehousing

De acordo com o *website* Solutions Review (King, 2021) existe um conjunto das melhores soluções de CDW, baseando-se nos produtos que melhor representam as condições atuais do mercado, de acordo com os seus utilizadores. Estes foram o *website* e as tecnologias selecionadas por recomendação da empresa na qual a presente dissertação se encontra associada. As 6 tecnologias selecionadas para um estudo inicial foram, respetivamente:

- Amazon Redshift: é um CDW totalmente gerido que permite que os clientes escalem de algumas centenas de gigabytes para um petabyte ou mais. A solução permite que os utilizadores carreguem qualquer conjunto de dados e realizem consultas de análise de dados. Independentemente do tamanho do conjunto de dados, o Redshift oferece desempenho de *queries* rápido utilizando ferramentas familiares baseadas em SQL e aplicações de inteligência de negócios;
- Google BigQuery: a Google oferece um CDW corporativo totalmente gerido para análise por meio do seu produto BigQuery. A solução não tem servidor e permite que as organizações analisem quaisquer dados criando um DW lógico. O BigQuery captura dados em tempo real utilizando um recurso de ingestão de *streaming* e é desenvolvido

com base no Google Cloud Platform. O produto também oferece aos utilizadores a capacidade de partilhar *insights* por meio de conjuntos de dados, *queries* e relatórios;

- IBM Db2 Warehouse: é um DW pré-configurado e gerido pelo cliente que é executado em *clouds* privadas, *clouds* privadas virtuais e outras infraestruturas suportadas por *containers*. O Db2 também oferece uma implementação flexível para que os utilizadores possam escrever aplicações uma vez e movê-los para o local certo com o mínimo ou nenhuma alteração necessária. Outros recursos importantes incluem, mas não estão limitados ao processamento rápido de *queries*, compatibilidade com Db2, PDA e Oracle e um mecanismo Apache Spark incorporado;
- Azure Synapse: é um serviço de análise que inclui integração de dados, armazenamento de dados corporativos e análise de Big Data. A solução permite que os utilizadores consultem dados utilizando recursos dedicados ou sem servidor. Ele oferece uma experiência unificada para ingerir, explorar, preparar, gerir e fornecer dados para Business Intelligence (BI) e ML. O Synapse também oferece recursos avançados de segurança e privacidade, como segurança em nível de coluna e linha;
- Snowflake: oferece um CDW que carrega e otimiza dados de praticamente qualquer fonte, estruturada e não estruturada, incluindo JSON, Avro e XML. O Snowflake oferece amplo suporte para SQL padrão e os utilizadores podem fazer atualizações, exclusões, funções analíticas, transações e junções complexas como resultado. A ferramenta não requer gestão nem infraestrutura. O mecanismo de base de dados colunar usa otimizações avançadas para processar dados, processar relatórios e executar análises;
- SAP Data Warehouse Cloud: é um serviço de DW criado na base de dados SAP HANA Cloud. Ele conecta dados em repositórios locais e repositórios multi-nuvem em tempo real, preservando o contexto de negócios. O produto também permite que os utilizadores modelem, visualizem e compartilhem dados num ambiente governado. Inclui modelos de dados pré-construídos, visualizações semânticas de dados de aplicativos SAP e lógica de transformação que também utiliza a experiência do fornecedor do seu ecossistema de parceiros.

3. SELEÇÃO DAS TECNOLOGIAS DE CLOUD DATA WAREHOUSE

Selecionar a tecnologia de CDW ideal pode ser uma tarefa complexa pois são várias as opções disponíveis no mercado e cada fornecedor tem os seus benefícios e desvantagens para um certo consumidor. Embora os principais fornecedores de CDW ofereçam funcionalidades aparentemente semelhantes em termos de escalabilidade, flexibilidade, confiabilidade, segurança, etc., os produtos atendem a diferentes casos de uso e têm uma estrutura de custos diferente. O objetivo do presente capítulo é a exposição do estudo realizado sobre quatro diferentes tecnologias de CDW, em termos de características inerentes às mesmas, através de uma revisão de artigos e informações. Isto será a base de conhecimento para posteriormente inicializar a utilização destas tecnologias e criar um ambiente de CDW em cada uma delas. Neste sentido, as quatro tecnologias que irão ser estudadas são, respetivamente, o Snowflake, o Google BigQuery, o Amazon Redshift e o Azure Synapse.

3.1 Introdução às Tecnologias Seleccionadas

De seguida são apresentadas, de uma forma mais pormenorizada, as quatro tecnologias de CDW seleccionadas para o desenvolvimento da presente dissertação, tendo sido estas seleccionadas pela empresa com a qual a presente dissertação se encontra associada. Esta seleção em específico justifica-se pelo facto de estes serem, de momento, uns dos quatro CDW mais utilizados no mercado, bem como dos que mais despertam dúvidas ao público, podendo, desta forma, tornar o alcance dos resultados do *benchmark* maior, e trazer uma maior dimensão e utilidade ao estudo.

3.1.1 Snowflake

Desenvolvido em 2012, o Snowflake é um Software as a Service (SaaS) que fornece uma plataforma única para DW, *data lakes*, *data engineering*, *data science*, desenvolvimento de aplicações de dados e partilha e consumo seguros de dados em tempo real. É um serviço totalmente gerido, ou seja, os seus utilizadores não se vão preocupar com nenhum trabalho de *back-end*, como a instalação do servidor, manutenção, entre outros. O Snowflake apresenta recursos prontos para uso, como separação de armazenamento e computação,

computação escalável em tempo real, partilha de dados, clonagem de dados e suporte a ferramentas de terceiros para lidar com as necessidades exigentes das empresas. Adicionalmente, como o Snowflake é independente da *cloud*, ele é executado em qualquer uma das três principais *clouds*, sendo essas a Amazon Web Services (AWS), a Google Cloud Platform (GCP) e o Azure, e suporta a versão padrão mais comum do SQL, a versão ANSI.

Depois de os dados serem carregados no Snowflake, os engenheiros de dados podem utilizar a lógica de negócios para os transformar e modelar, de uma maneira a que faça sentido para a sua empresa. Essas informações são depois utilizadas para alimentar relatórios e painéis para que os *stakeholders* do negócio possam tomar decisões importantes (Kline, 2022).

3.1.2 Big Query

O BigQuery é uma Plataforma as a Service (PaaS), um CDW empresarial totalmente gerido e com um mecanismo de *queries* integrado, que ajuda o utilizador a gerir e a analisar os seus dados com recursos integrados, como ML, *geospatial analysis* e BI. Foi lançado pela primeira vez em 2010 como parte da Google Cloud Platform e foi uma das primeiras soluções de armazenamento de dados disponíveis no mercado. A arquitetura sem servidor do BigQuery permite que se utilize *queries* SQL para responder às maiores perguntas da organização, sem se preocupar com a gestão de infraestrutura. Ele foi projetado para analisar dados na ordem de bilhões de linhas, e é compatível com o Google Standard SQL, apesar de o Legacy SQL está também disponível. O mecanismo de análise distribuída e escalável do BigQuery permite consultar *terabytes* em segundos e *petabytes* em minutos. Desta forma, o BigQuery maximiza a flexibilidade separando o mecanismo de computação que analisa os seus dados das opções de armazenamento. Adicionalmente, ele permite que sejam lidos dados de fontes externas e o *streaming* oferece suporte a atualizações contínuas de dados (Kline, 2021).

3.1.3 Amazon Redshift

Em 2013, a AWS, como um PaaS, revolucionou o setor de DW ao lançar o Amazon Redshift, o primeiro CDW de nível empresarial, em escala de *petabytes*. O Amazon Redshift tornou simples e económico analisar com eficiência grandes volumes de dados utilizando as

ferramentas de BI existentes. Este lançamento foi um salto significativo em relação às soluções tradicionais de armazenamento de dados *on-premise*, que eram caras, não elásticas e exigiam experiência significativa para ajustar e operar (Pandis, I., 2021). O mecanismo do Amazon Redshift é um sistema de gestão de base de dados e processamento de *queries* projetado para dar suporte à carga de trabalho de análises. Ele é baseado no PostgreSQL, portanto, a maioria das aplicações SQL existentes funcionarão com apenas alterações mínimas. O Redshift é um serviço nativo da AWS, criado para funcionar em conjunto com outras tecnologias da AWS e projetado para ser um *hub* central para conectar essas outras ofertas (Kline, 2022).

3.1.4 Azure Synapse

Com vários recursos para integração de dados, DW e análise de *big data*, o Azure Synapse Analytics, anteriormente conhecido por Azure SQL Data Warehouse, é um PaaS oferecido pela Microsoft, que oferece uma plataforma de dados elásticos geridos para análise de dados públicos, operacionais e históricos. Esta é uma oferta relativamente nova da Microsoft, tendo sido lançada oficialmente apenas no final de 2020, com o objetivo de ser uma plataforma única onde as empresas podem recolher e consolidar dados e usar SQL para fins de análise, e que usa a linguagem Transact-SQL (T-SQL). Os seus clientes podem adaptar a sua infraestrutura de análise às suas necessidades de processamento, com opções de preços dedicadas e sem servidor, baseadas no uso. O Synapse pode também pausar e retomar a cobrar o tempo de computação, o que significa que apenas o armazenamento é cobrado durante o tempo de pausa. Desta forma, ele alcança um bom equilíbrio em termos de configurabilidade e simplicidade, de uma forma que é fácil de administrar e flexível para lidar com quase qualquer padrão de uso. Além dos casos de uso de análise, o Synapse foi projetado para atuar como um *hub* central para conectar ofertas adicionais do Azure. Assim, o Synapse integra-se ao Apache Spark de modo a lidar com cargas de trabalho de *streaming*, inteligência artificial e ML, para além de cargas de trabalho convencionais de SQL e BI (Kline, 2021).

3.2 Comparação das Tecnologias de Cloud Data Warehouse

No seguinte capítulo será apresentado o estudo das quatro tecnologias de CDW selecionadas para o presente estudo, relativamente a um conjunto selecionado de características relevantes, respetivamente arquitetura, escalabilidade, segurança, suporte de dados, e ainda relativamente ao seu modelo de preços. Foram selecionadas especificamente estas características visto estes serem dos critérios mais relevantes para o consumidor geral, segundo a pesquisa e estudo realizados ao longo da presente dissertação e segundo a opinião da empresa na qual a presente dissertação se encontra associada.

3.2.1 Arquitetura

O Snowflake tem as camadas de armazenamento e de processamento de computação completamente separadas, tornando extremamente fácil aumentar ou diminuir o DW conforme necessário. Os nós de computação ou *clusters* no Snowflake são conhecidos como depósitos individuais, onde são armazenadas partes de cada conjunto de dados. As micropartições são utilizadas para otimizar todos os dados e compactá-los num formato de armazenamento colunar, para que possam ser mantidos no armazenamento em *cloud*. Por fim, o Snowflake é inteiramente baseado em ANSI SQL, portanto, a barreira de entrada é extremamente baixa. Como o Snowflake é independente da *cloud*, ele é executado em qualquer uma das três principais *clouds*, sendo essas a AWS, a GCP e o Azure (Kline, 2022). Na figura 6 está representada a arquitetura do Snowflake previamente descrita.

O Google BigQuery é muito semelhante ao Snowflake, pois é *serverless*, separa as camadas de armazenamento e de processamento de computação e também é baseado em ANSI SQL. No entanto, a sua arquitetura é bastante diferente. O BigQuery utiliza um vasto conjunto de serviços orientados por tecnologias de infraestrutura específicas do Google, respetivamente DREMEL, COLOSSUS, JUPITER e BORG. A computação no Google BigQuery é realizada utilizando o DREMEL, que é um grande *cluster* de computação utilizado para executar *queries* SQL. Semelhante ao Snowflake, o BigQuery compacta dados num formato colunar para armazenar dados no COLOSSUS, que é o sistema de armazenamento global da Google. Posteriormente, o BigQuery utiliza a rede JUPITER do Google para mover os seus dados rapidamente de um local para outro. Toda a alocação e orquestração de recursos de

hardware no BigQuery é feita por meio do BORG, que é o precursor da Google para o Kubernetes (Kline, 2021). Na figura 7 está representada a arquitetura do BigQuery previamente descrita.

O Redshift é funcionalmente semelhante ao Snowflake, pois aproveita o Massively Parallel Processing (MPP), mas estruturalmente é muito diferente. Em primeiro lugar, o Redshift só pode ser executado na AWS, não é agnóstico em *cloud* como o Snowflake, portanto, há imediatamente menos flexibilidade. Os recursos de armazenamento e computação no Redshift são fortemente conectados e não são naturalmente sem servidor como o Snowflake. No entanto, é importante observar que o Redshift oferece um recurso não conectado, por meio dos seus nós RA3, além de uma opção sem servidor. Adicionalmente, o Redshift aproveita *clusters* como o Snowflake, e os nós dentro de um *cluster* são particionados em fatias, onde cada fatia representa uma porção alocada do disco e do espaço de memória de um nó individual. Tudo isso é gerido por um nó líder que lida com a comunicação, compila o código e atribui uma parte dos dados a cada nó de computação. Por fim, o armazenamento no Redshift é duplicado do S3 e os dados podem ser compactados e armazenados num formato colunar semelhante ao Snowflake. (Kline, 2022). Na figura 8 está representada a arquitetura do Redshift previamente descrita.

O Azure Synapse foi desenvolvido exclusivamente para ser executado no Azure Cloud. As camadas de armazenamento e de computação do Synapse são também separadas, e, mais uma vez, a plataforma é compatível com ANSI SQL. No centro do Azure Synapse está o estúdio Synapse, que é a interface do utilizador para monitorizar recursos, executar tarefas, escrever código e gerir o acesso do utilizador. Enquanto o Snowflake usa depósitos virtuais, o Azure Synapse oferece o que é conhecido como Pools SQL Dedicados, Pools SQL Sem Servidor e Pools Spark. O poder de computação em Pools SQL Dedicados é determinado por nós de computação conhecidos como Data Warehouse Units (DWU). O Synapse então utiliza o MPP para distribuir *queries* em vários nós de computação (Kline, 2021). Na figura 9 está representada a arquitetura do Azure Synapse.

Assim, é possível concluir que apesar de existirem bastantes parecenças nas arquiteturas das quatro tecnologias, estas também diferem de várias formas. O Google BigQuery é muito semelhante ao Snowflake, pois é *serverless*, tem o armazenamento separado da computação e também é baseado em ANSI SQL. No entanto, a sua arquitetura é

bem diferente, pois este usa um vasto conjunto de serviços orientados por tecnologias de infraestrutura específicas do Google. O Redshift é também funcionalmente semelhante ao Snowflake, pois aproveita o MPP, mas é estruturalmente muito diferente, como por exemplo só poder ser executado na AWS e os recursos de armazenamento e computação no Redshift serem fortemente conectados. Por fim, no Azure Synapse o armazenamento e a computação são também separados, e a plataforma é compatível com ANSI SQL, no entanto, também apresenta outras diferenças, mencionadas ao longo do texto.

3.2.2 Escalabilidade

A escalabilidade é a capacidade de escalar recursos tanto horizontalmente, ou seja, provisionamento de outras instâncias num *cluster*, como verticalmente, ou seja, aumento de recursos de uma instância. Essa capacidade é refletida em tempo real de modo a não gerar o mal funcionamento do serviço. Na tabela 1 estão apresentadas as principais características sobre escalabilidade relativamente a cada uma das tecnologias.

Snowflake	O Snowflake possui um recurso exclusivo de dimensionamento automático e suspensão automática para iniciar e parar armazéns durante períodos ociosos e ocupados. Apesar de não ser possível redimensionar nós individuais no Snowflake, este oferece aos seus utilizadores a capacidade de redimensionar <i>clusters</i> num único clique e, como cada depósito está no seu próprio <i>cluster</i> de computação, as cargas de trabalho podem ser isoladas individualmente para permitir <i>queries</i> simultâneas ilimitadas. No entanto, certas <i>queries</i> podem ser ineficientes e difíceis de provisionar no Snowflake, pois os nós individuais não podem ser ajustados e a única maneira de obter nós maiores é ter <i>warehouses</i> maiores (Kline, 2022).
Google BigQuery	De forma semelhante ao Snowflake, o BigQuery fornece automaticamente os seus recursos de computação adicionais, conforme sejam necessários, ou seja, permitem que se aumente e diminua automaticamente com base na procura. Isto dá origem à obtenção de alta escalabilidade de dados que são executados em tempo real, tanto vertical como horizontalmente, e até

	<i>petabytes</i> de dados. No entanto, com o BigQuery, há um limite padrão de 100 utilizadores simultâneos (Kline, 2021).
Amazon Redshift	O escalonamento automático do Redshift é limitado e pode levar de vários minutos a várias horas para ajustar os <i>clusters</i> , pois os nós individuais precisam de ser adicionados e removidos manualmente. A dificuldade de dimensionar no Redshift dependem bastante do tamanho dos <i>clusters</i> e da quantidade de dados (Kline, 2022).
Azure Synapse	As Pools SQL Sem Servidor e as Pools Spark têm dimensionamento automático por padrão, mas as Pools de servidores SQL Dedicados precisam de ser ajustadas manualmente pelo utilizador, pois não há recurso de suspensão automática como, por exemplo, o Snowflake (Kline, 2021).

Tabela 1 - Escalabilidade nas Tecnologias de CDW

Relativamente a capacidades de escalabilidade, pode-se concluir que o Snowflake e o BigQuery são bastante semelhantes, enquanto o Redshift e o Synapse têm menos recursos de escalabilidade quando comparado aos dois primeiros.

3.2.3 Segurança dos Dados

O local onde os dados residem e as políticas de segurança que cada fornecedor oferece podem ser uma parte fundamental do processo de seleção de fornecedor de serviços de *cloud*. Na tabela 2 estão apresentadas as principais características sobre segurança dos dados relativamente a cada uma das tecnologias.

Snowflake	Uma das melhores facetas da segurança do Snowflake é o Role-based Access Control (RBAC), utilizado para gerir utilizadores e privilégios, bem como a utilização do Multi Factor Authentication (MFA), de modo a aumentar a segurança da conta. Como o Snowflake opera em <i>clouds</i> , este pode usar Virtual Private Networks (VPN) como o Azure Private Link ou o AWS PrivateLink. Adicionalmente, o Snowflake também criptografa
------------------	---

	automaticamente todos os arquivos armazenados nos estágios internos, para carregar e descarregar dados automaticamente (Kline, 2022).
Google BigQuery	O Snowflake fornece criptografia automática para dados em repouso, no entanto, não fornece permissões granulares para colunas, mas sim apenas para <i>schemas</i> , tabelas, <i>views</i> , <i>procedures</i> e outros objetos. O BigQuery oferece segurança ao nível de colunas, bem como permissões em conjuntos de dados, tabelas individuais, <i>views</i> e controlos de acesso a tabelas. Como o BigQuery é uma oferta nativa da Google, também se pode aproveitar outros serviços do Google Cloud que têm segurança e autenticação integradas ao BigQuery, facilitando muito as integrações (Kline, 2021).
Amazon Redshift	Além de lidar com toda a segurança de infraestrutura e <i>hardware</i> , o Redshift também gere a segurança em torno do provisionamento de recursos. A gestão de acesso é feita pelo AWS Identity and Access Management (IAM). Somente utilizadores com as credenciais apropriadas podem criar, configurar e excluir <i>clusters</i> . O acesso a qualquer recurso do Redshift é tratado pelos privilégios da conta da AWS e os grupos de segurança podem ser atribuídos a instâncias de <i>cluster</i> específicas para conceder acesso de entrada. Os <i>clusters</i> podem também ser executados na AWS Virtual Private Cloud para proteger ainda mais o acesso (Kline, 2022).
Azure Synapse	O Synapse também criptografa automaticamente os dados em repouso e oferece o RBAC, utilizado para gerir utilizadores e privilégios bem como a utilização do MFA, de modo a aumentar a segurança da conta. Este pode também usar VPN como o Azure Private Link, sendo que, em contraste, o Snowflake pode também utilizar o AWS PrivateLink. Adicionalmente, o Synapse também possui um recurso conhecido como Microsoft Defender que monitoriza recursos e fornece recursos integrados na deteção de ameaças, de modo a se detetar explorações e mitigar atividades prejudiciais (Kline, 2021).

Tabela 2 - Segurança nas Tecnologias de CDW

Relativamente à segurança de dados, é possível afirmar que qualquer uma das tecnologias apresenta boas características e métodos de segurança, havendo poucas diferenças de qualidade entre cada uma delas.

3.2.4 Suporte de Dados

Na tabela 3 estão apresentadas as principais características relativamente ao suporte de dados oferecido em cada uma das tecnologias.

Snowflake	O Snowflake é compatível com dados semi-estruturados (JSON, Avro, Orc, CSV – Comma-separated Values, Parquet) e dados estruturados. O Snowflake também lançou suporte para dados não estruturados no final de 2021, permitindo que os desenvolvedores aproveitem ainda mais linguagens de desenvolvimento (ex: Python, Java, Scala). Isso dá ao Snowflake ainda mais flexibilidade como plataforma de dados, pois pode ser tratado tanto como <i>data lake</i> como DW (Kline, 2022).
Google BigQuery	O BigQuery também suporta dados semi-estruturados (JSON, Avro, Orc, CSV, Parquet) e dados estruturados (Kline, 2021).
Amazon Redshift	Tal como o Snowflake e o BigQuery, o Redshift também oferece suporte a dados estruturados e semi-estruturados. Adicionalmente, o Redshift também suporta PartiQL, que é uma linguagem de <i>query</i> projetada para processar dados semi-estruturados em bases de dados relacionais com mais eficiência. No entanto, o Redshift não oferece suporte a dados não estruturados (Kline, 2022).
Azure Synapse	No Synapse suporta também tipos de dados estruturados e semi-estruturados. No entanto, para além do SQL, o Synapse permite que os desenvolvedores utilizem várias linguagens de programação como Python, SQL, Java, Scala, .NET e R. integração com o Azure Data Lake e o Delta Lake, o que o torna a escolha ideal para dados não estruturados (Kline, 2021).

Tabela 3 - Suporte de dados nas Tecnologias de CDW

De uma forma geral, as quatro tecnologias apresentam um suporte de dados semelhante, tendo sempre suporte a dados estruturados e semi-estruturados. No entanto, adicionalmente, o Snowflake e o Azure Synapse também apresentam suporte a dados não estruturados.

3.2.5 Modelo de Preços

Embora nunca deva ser o fator único ou mais importante, não há como negar que o custo desempenhará um grande papel na decisão de qual fornecedor de serviços de *cloud* se deve escolher, visto que muitas empresas procuram migrar para a *cloud* também para reduzir os seus custos de produção. Na tabela 4 estão apresentadas as principais características relativamente aos custos de uso de cada uma das tecnologias.

Snowflake	<p>O modelo de preços no Snowflake é baseado no uso individual do DW. Todos os DW virtuais vêm em vários tamanhos diferentes como X-Small, Small, Medium, Large, X-Large, etc, sendo que um DW X-Small utiliza apenas um <i>cluster</i>. Para cada aumento no tamanho do DW, o número de <i>clusters</i> duplica junto com o custo. Para um DW X-Small, o preço começa em aproximadamente 0,0003 créditos por segundo. Dependendo do nível do Snowflake, o custo por crédito pode variar substancialmente, mas o preço <i>on-demand</i> do Snowflake Standard Edition é de US\$ 2. O maior DW, ou seja, o 6X-Large, custa 512 créditos por hora ou 0,1422 créditos por segundo. No Snowflake, os clientes podem comprar créditos antecipadamente ou pagar consoante uso. A compra antecipada de créditos economiza uma quantia substancial de dinheiro, pois o Snowflake tem ofertas especiais quando os clientes assinam contratos mais longos e pagam antecipadamente. Embora os custos de crédito possam variar substancialmente no Snowflake, dependendo da camada de negócios, os custos de armazenamento no Snowflake são relativamente simples. Os utilizadores <i>on-demand</i> pagam uma taxa fixa de US\$ 40 por mês para cada TB de dados e os clientes iniciais pagam US\$ 23 por TB de dados armazenados (Kline, 2022).</p>
------------------	--

<p>Google BigQuery</p>	<p>Por outro lado, o BigQuery cobra pelo número de <i>bytes</i> verificados ou lidos. O BigQuery oferece também preços <i>on-demand</i> e preços fixos. O preço <i>on-demand</i> cobra pelo número de <i>bytes</i> processados numa determinada <i>query</i>, a uma taxa de US\$ 5 por TB, apesar que o primeiro TB de dados processados por mês é totalmente gratuito. Com o modelo de preços fixo do BigQuery, o utilizador compra <i>slots</i> (CPUs virtuais) ou recursos dedicados para executar suas <i>queries</i>. O custo mensal de 100 slots é de cerca de US\$ 2.000, que pode ser reduzido para US\$ 1.700 com um compromisso anual. Os custos de armazenamento do Snowflake e do BigQuery são relativamente baratos, onde o BigQuery cobra US\$ 20 por TB para armazenamento ativo e US\$ 10 por TB para armazenamento inativo (Kline, 2021).</p>
<p>Amazon Redshift</p>	<p>A estrutura de preços do Redshift é um pouco mais complexa em comparação com as restantes pois há mais opções especificamente em torno dos tipos de nós. O Redshift oferece dois tipos principais de nós, DC2 e RA3. Os nós DC2 são fortemente conectados ao armazenamento. Com os nós RA3, o armazenamento e a computação são geridos de forma independente. O preço de uma instância DC2 varia de US\$ 0,25 por hora para o menor tamanho, a US\$ 4,80 por hora para o maior tamanho. Por outro lado, as instâncias RA3 começam em US\$ 1,086 por hora para o menor tamanho, e US\$ 13,04 para o maior. O Redshift também oferece uma opção sem servidor para utilizadores que não desejam provisionar e dimensionar <i>hardware</i>. Com essa opção, o Redshift aumenta ou diminui automaticamente para atender aos requisitos de cargas de trabalho analíticas e é encerrado durante períodos de inatividade. Relativamente ao consumo, este é calculado por minuto com base nas horas de Redshift Processing Unit (RPU), em que o preço de uma RPU é de US\$ 0,45 por hora. Um aspeto importante a ser observado sobre o Redshift é que a AWS oferece grandes descontos para compromissos de contrato de longo prazo. Quanto ao armazenamento no Redshift, este é um pouco mais caro que o Snowflake ou o BigQuery, sendo a US\$ 24 por TB. No entanto, os dados também podem ser armazenados no S3 e consultados diretamente usando o Redshift Spectrum. O custo de</p>

	armazenamento no S3 começa em US\$ 23 por TB, mas pode ser reduzido dependendo da prioridade na qual os dados precisam de ser acedidos, pois os dados podem ser mantidos em diferentes camadas de armazenamento no S3. Isso torna o Redshift substancialmente mais barato que, por exemplo, o Snowflake, em alguns casos (Kline, 2022).
Azure Synapse	Os modelos de preço com o Synapse são também um pouco mais complexos, pois há também mais opções em comparação com o Snowflake. Os recursos de computação ou DWUs no Azure começam em US\$ 1,51 por hora para o menor tamanho e podem ir até US\$ 453 para o maior tamanho. Como o Snowflake, porém, esses custos podem ser drasticamente reduzidos com a compra de capacidade reservada. Por outro lado, as Pools do SQL Serverless cobram uma taxa fixa de US\$ 5,65 para cada TB de dados processados. Os custos de armazenamento no Azure são também um pouco mais caros, custando cerca de US\$ 26 por TB por mês. Uma coisa importante a ser observada é que o Snowflake cobra por computação por minuto e o Azure Synapse cobra por hora. Isso significa que, se a execução da consulta no Snowflake levar 3 minutos, os usuários pagarão por 3 minutos de computação. Por outro lado, se o Azure Synapse estiver ativo por 30 minutos, os utilizadores ainda serão forçados a pagar por uma hora inteira. É por isso que parece haver uma discrepância tão grande no modelo de preços de ambas as soluções (Kline, 2021).

Tabela 4 - Modelo de Preços das Tecnologias de CDW

Ao longo do texto foram especificadas as demais características relativas aos modelos de preços de cada uma das tecnologias, no entanto, visto que cada tecnologia apresenta um modelo que funciona de forma diferente, só é possível compará-los diretamente se se souber exatamente as intenções de uso e de necessidade do utilizador em causa.

4. CONSTRUÇÃO DO AMBIENTE DE CLOUD DATA WAREHOUSE

De seguida é apresentado o estudo realizado sobre as quatro diferentes tecnologias de CDW, bem como o seu processo de preparação e execução. Para além da análise características das tecnologias selecionadas realizada no capítulo anterior, com a presente dissertação é pretendido elaborar um *benchmark* do desempenho das diversas tecnologias, através da criação de um ambiente de CDW em cada uma, e avaliando um conjunto de métricas pré-definidas.

Para tal, foi utilizado um *dataset* reconhecimento nesta área, o Star Schema Benchmark (SSB). Ao longo do presente capítulo serão especificadas as características do *dataset* do SSB e especificadas as configurações escolhidas para a criação de cada um dos ambientes de CDW. Adicionalmente, será exemplificado o processo de criação dos ambientes de CDW, bem como descritas e explicadas as métricas que irão ser medidas na realização deste *benchmark*.

4.1 Dataset: Star Schema Benchmark

Numa fase inicial de desenvolvimento da dissertação foi proposto, pela empresa associada à mesma, a utilização do *dataset* derivado do reconhecido padrão da indústria TPC Benchmark™ H (TPC-H). O TPC-H é uma referência de suporte à decisão que consiste num conjunto de *queries ad-hoc* orientadas para negócios e em modificações de dados simultâneas. Este *benchmark* ilustra sistemas de suporte à decisão que examinam grandes volumes de dados, executam *queries* com alto grau de complexidade e dão respostas a questões críticas de negócios (Indicative Team, 2021).

No entanto, o TPC-H *benchmark* tem sido fortemente criticado por não aderir estritamente aos princípios do modelo de Ralph Kimball, visto ele apoiar a ideia de que um sistema de apoio à decisão deve utilizar um esquema em estrela. Como resultado, O'Neil et al. (2009) propôs um conjunto de modificações ao TPC-H, ao qual eles denominaram de Star Schema Benchmark (SSB). Para o desenvolvimento da presente dissertação foi então utilizado o *dataset* deste novo SSB, de modo que o estudo seja o mais apropriado à realidade possível.

O SSB foi projetado para testar a otimização do esquema em estrela para resolver os problemas descritos no TPC-H, com o objetivo de medir o desempenho de produtos de base

de dados e testar uma nova estratégia de materialização. Ele é um *benchmark* simples que consiste em quatro conjuntos de *queries*, quatro dimensões e uma tabela de factos. O modelo relacional do SSB é baseado no modelo do TPC-H, com melhorias que implementam um esquema em estrela puro tradicional e permite a compactação de colunas e tabelas. Tais melhorias e alterações foram especificadas por O'Neil et al. (2009) e serão discriminadas ao longo do presente capítulo.

4.1.1 Modelo Relacional

Foram realizadas várias modificações ao modelo do TPC-H de modo a traduzi-lo num esquema em estrela mais eficiente. A figura 4 ilustra o modelo relacional do TPC-H benchmark, e a figura 5 ilustra o modelo relacional do SSB com as devidas alterações, composto por uma tabela de factos, a LINEORDER e quatro tabelas de dimensão, a CUSTOMER, PART, SUPPLIER e DATE.

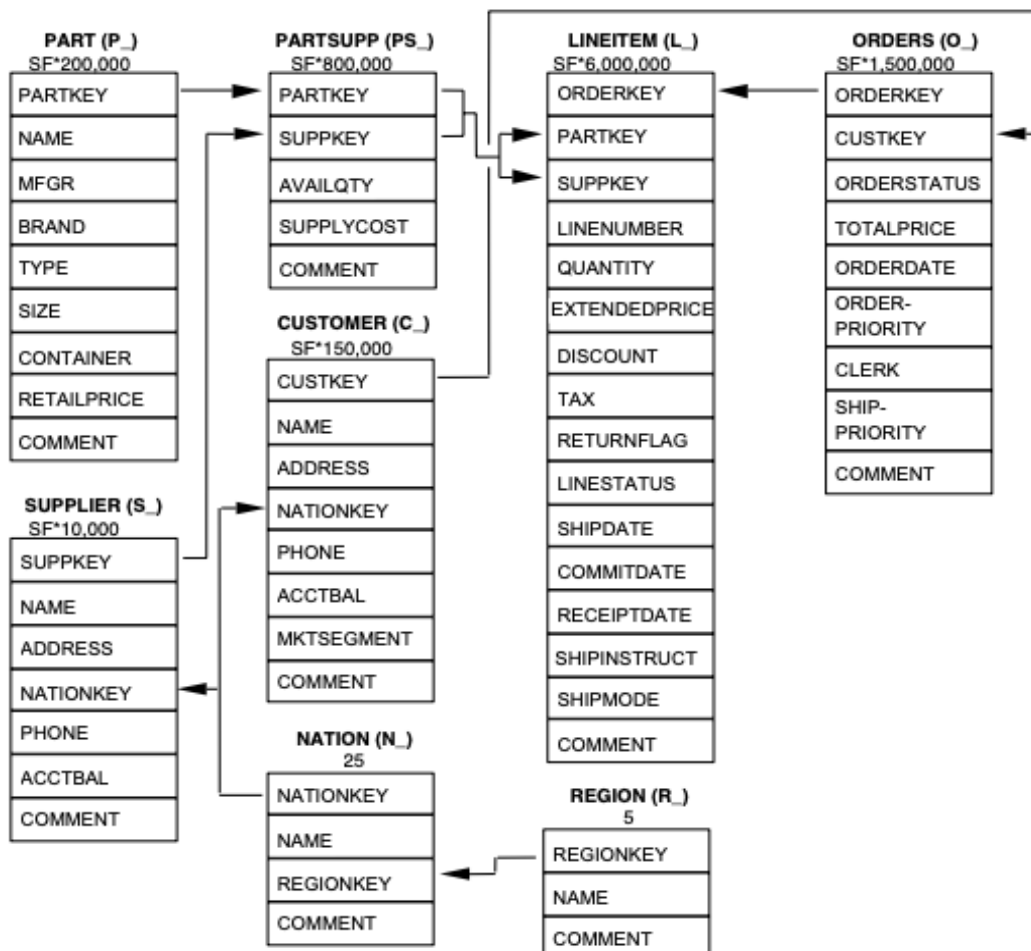


Figura 4 - Modelo Relacional TPC-H Benchmark

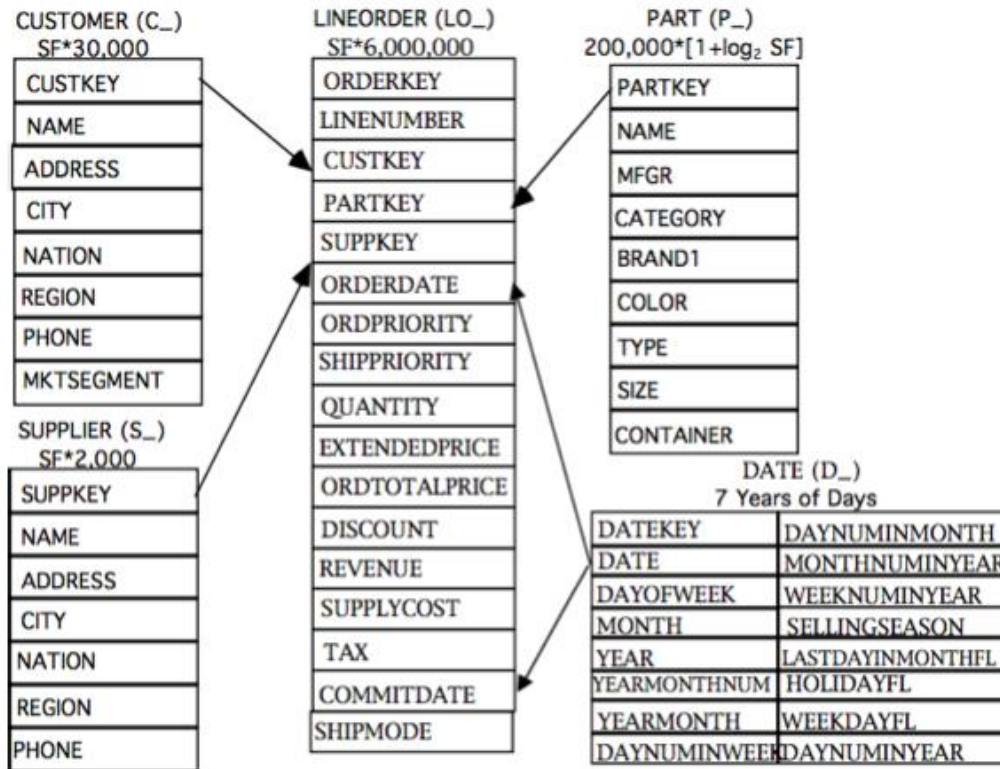


Figura 5 - Modelo Relacional SS Benchmark

Em comparação ao TPC-H, são a seguir discriminadas as mudanças aplicadas ao seu *schema*, todas baseadas no documento desenvolvido por O'Neil et al. (2009), de forma que este se transforme num esquema em estrela eficiente do SSB:

- As tabelas LINEITEM e ORDERS do TPC-H foram combinadas numa tabela de fatos de vendas, denominada de LINEORDER. Essa desnormalização é padrão nos DW, conforme explicado por Kimball, e torna muitos JOINS desnecessários em *queries* comuns;
- A tabela PARTSUPP foi descartada pois esta pertenceria a um *data mart* diferente das informações ORDERS e LINEITEM. Isso ocorre pois o PARTSUPP possui granularidade temporal diferente, o que significa que atualizações que adicionem novas linhas ao longo do tempo a LINEORDER não iriam adicionar novas linhas a PARTSUPP, pois esta está congelada no tempo;
- As tabelas NATION e REGION foram descartadas, pois estas transformariam o esquema em estrela num esquema em floco de neve, quando unidas às dimensões CUSTOMER,

PART e SUPPLIER. No entanto, informações presentes nas mesmas foram adicionadas a outras dimensões; e

- Foi adicionada a dimensão DATE, como é padrão para um DW de vendas, composta por dados no intervalo de 7 anos, entre 1992 e 1997, inclusive. Ao contrário das tabelas descartadas NATION e REGION, esta tabela possui um grande número de atributos valiosos em *queries*, como DAYOFWEEK, MONTH, YEAR, etc.

4.1.2 Estrutura das Tabelas e Alterações Aplicadas

As tabelas 5 a 9 definem a estrutura de cada tabela presente no esquema do SSB. Adicionalmente às colunas presentes em cada tabela, são também especificadas as modificações realizadas em relação aos dados presentes na base de dados do TPC-H, de modo a criar o esquema em estrela adequado, conforme a figura 5. Tal significa que nos CDWs foi necessário o carregamento dos dados em tabelas conforme o modelo relacional do TPCH-H, ilustrado na figura 4, para, posteriormente, se proceder à evolução para o modelo relacional do SSB. Todas as alterações e justificações para a seleção das colunas para cada tabela podem ser consultadas com mais detalhe no documento de O’Neil (2009), no qual estas foram baseadas. A estrutura das tabelas do *dataset* TPC-H pode ser consultada no Anexo A – Estrutura das Tabelas do Schema TPC-H, de modo a auxiliar o entendimento e dimensão das alterações realizadas. Posteriormente, durante a criação dos ambientes de CDW em cada uma das tecnologias selecionadas foi necessária a criação, por autoria própria, de *stored procedures* com código SQL que efetivamente aplicassem estas alterações aos dados em causa. Esse processo será pormenorizado num capítulo posterior.

Tabela FACT_LINEORDER

Nome da Coluna	Tipo de Dados	Alterações / Comentários
LO_ORDERKEY	Integer	Chave Primária / O_ORDERKEY de ORDERS
LO_LINENUMBER	Integer	Chave Primária / L_LINENUMBER de LINEITEM
LO_CUSTKEY	Integer	O_CUSTKEY de ORDERS
LO_PARTKEY	Identifier	L_PARTKEY de LINEITEM
LO_SUPPKEY	Integer	L_SUPPKEY de LINEITEM
LO_ORDERDATE	Identifier	O_ORDERDATE de ORDERS
LO_ORDERPRIORITY	Variable text, size 15	O_ORDERPRIORITY de ORDERS
LO_SHIPPRIORITY	Variable text, size 1	O_SHIPPRIORITY de ORDERS
LO_QUANTITY	Integer	L_QUANTITY de LINEITEM
LO_EXTENDEDPRICE	Decimal	L_EXTENDEDPRICE de LINEITEM
LO_ORDTOTALPRICE	Integer	O_TOTALPRICE de ORDERS
LO_DISCOUNT	Decimal	L_DISCOUNT de LINEITEM
LO_REVENUE	Decimal	Formula: $L_EXTENDEDPRICE * (100 - L_DISCOUNT) / 100$ de LINEITEM
LO_SUPPLYCOST	Decimal	PS_SUPPLYCOST de PARTSUPP
LO_TAX	Integer	L_TAX de LINEITEM
LO_COMMITDATE	Date	L_COMMITDATE de LINEITEM
LO_SHIPMODE	Variable text, size 10	L_SHIPMODE de LINEITEM

Tabela 5 - Estrutura da Tabela FACT_LINEORDER

De forma a obter este resultado de dados, foi necessário proceder às seguintes ligações de tabelas do modelo do TPC-H:

- A ligação entre a tabela LINEITEM e a tabela ORDERS é feita através de um JOIN entre a O_ORDERKEY em ORDERS e a L_ORDERKEY em LINEITEM; e
- A ligação entre a tabela LINEITEM e a tabela PARTSUPP é feita através de um JOIN entre a PS_PARTKEY em PARTSUPP e a L_PARTKEY em LINEITEM.

Tabela DIM_PART

Nome da Coluna	Tipo de Dados	Comentários
P_PARTKEY	Identifier	Chave Primária / P_PARTKEY de PART
P_NAME	Variable text, size 22	P_NAME de PART encurtado
P_MFGR	Variable text, size 6	P_MFGR de PART
P_CATEGORY	Variable text, size 7	P_BRAND de PART
P_BRAND1	Variable text, size 9	Subdivisão do P_CATEGORY
P_COLOR	Variable text, size 60	P_NAME de PART
P_TYPE	Variable text, size 25	P_TYPE de PART
P_SIZE	Integer	P_SIZE de PART
P_CONTAINER	Variable text, size 10	P_CONTAINER de PART

Tabela 6 - Estrutura da Tabela DIM_PART

De forma a obter este resultado de dados, foi necessário proceder às seguintes alterações aos dados do modelo do TPC-H:

- A coluna P_NAME da tabela PART foi encurtada, visto que esta é irrealisticamente longa, tendo 55 bytes e cinco cores concatenadas. No SSB a coluna P_NAME tem 22 bytes e duas cores concatenadas;
- A *string* 'Manufacturer' presente nos valores de P_MFGR foi substituída pela *string* 'MFGR';
- O nome P_BRAND que existia no TPC-H foi alterado para P_CATEGORY NO SSB, já que o P_BRAND tem apenas 25 valores distintos, um número bastante pequeno. Foi ainda adicionada uma nova coluna, P_BRAND1, com 1000 valores, subdividindo cada P_CATEGORY em 40. Por exemplo, um valor P_BRAND1 como MFGR#5433 acumula o valor P_CATEGORY MFGR#54, que acumula o valor P_MFGR MFGR#5. Isto trouxe complexidade ao modelo; e
- O P_COLOR é constituído pelos valores presentes no P_NAME de TPC-H, desta vez já tendo as 5 cores concatenadas.

Tabela DIM_SUPPLIER

Nome da Coluna	Tipo de Dados	Comentários
S_SUPPKEY	Identifier	Chave Primária / S_SUPPKEY de SUPPLIER
S_NAME	Variable text, size 25	S_NAME de SUPPLIER
S_ADDRESS	Variable text, size 50	S_ADDRESS de SUPPLIER
S_CITY	Variable text, size 10	N_NAME de NATION + dígitos de 0 a 9
S_NATION	Variable text, size 15	N_NAME de NATION
S_REGION	Variable text, size 12	R_NAME de REGION
S_PHONE	Variable text, size 15	S_PHONE de SUPPLIER

Tabela 7 - Estrutura da Tabela DIM_SUPPLIER

De forma a obter este resultado de dados, foi necessário proceder às seguintes ligações de tabelas e alterações aos dados do modelo do TPC-H:

- A ligação entre a tabela SUPPLIER e as tabelas NATION e REGION é feita através de um JOIN entre a S_NATIONKEY em SUPPLIER do TPC-H e a N_NATIONKEY em NATION, e, posteriormente, entre a N_REGIONKEY em NATION e a R_REGIONKEY em REGION; e
- Foi adicionada a coluna S_CITY utilizando os primeiros 9 caracteres de N_NAME de NATION, seguido por um dígito entre 0-9, permitindo que esta tenha uma cardinalidade de 250, enquanto na tabela NATION tem apenas 25 valores.

Tabela DIM_CUSTOMER

Nome da Coluna	Tipo de Dados	Comentários
C_CUSTKEY	Identifier	Chave Primária / C_CUSTKEY de CUSTOMER
C_NAME	Variable text, size 25	C_NAME de CUSTOMER
C_ADDRESS	Variable text, size 50	C_ADDRESS de CUSTOMER
C_CITY	Variable text, size 10	N_NAME de NATION + dígitos de 0 a 9
C_NATION	Variable text, size 15	N_NAME de NATION
C_REGION	Variable text, size 12	R_NAME de REGION
C_PHONE	Variable text, size 15	C_PHONE de CUSTOMER
C_MKTSEGMENT	Variable text, size 10	C_MKTSEGMENT de CUSTOMER

Tabela 8 - Estrutura da Tabela DIM_CUSTOMER

- A ligação entre a tabela CUSTOMER e as tabelas NATION e REGION é feita através de um JOIN entre a C_NATIONKEY em CUSTOMER e a N_NATIONKEY em NATION, e, posteriormente, entre a N_REGIONKEY em NATION e a R_REGIONKEY em REGION;
- É adicionada a coluna C_CITY utilizando os primeiros 9 caracteres de N_NAME de NATION, seguido por um dígito entre 0-9, permitindo que esta tenha uma cardinalidade de 250, enquanto NATION tem apenas 25 valores.

Tabela DIM_DATE

Nome da Coluna	Tipo de Dados	Comentários
D_DATEKEY	Identifier	Chave Primária
D_DATE	Variable text, size 10	Data no formato AAAA-MM-DD
D_DAYOFWEEK	Variable text, size 10	Dia da semana
D_MONTH	Variable text, size 9	Nome completo do mês do ano
D_YEAR	Integer	Ano
D_YEARMONTHNUM	Integer	Junção do ano e mês, em numérico, exemplo: '199201', '199202'
D_YEARMONTH	Variable text, size 7	Junção do ano e mês, por extenso, exemplo: 'Jan1992', 'Fev1992'
D_DAYNUMINWEEK	Integer	Número do dia na semana, exemplo: se for domingo, então '1', se for segunda-feira, então '2'
D_DAYNUMINMONTH	Integer	Número do dia no mês
D_DAYNUMINYEAR	Integer	Número do dia no ano
D_MONTHNUMINYEAR	Integer	Número do mês no ano
D_WEEKNUMINYEAR	Integer	Número da semana no ano
D_LASTDAYINWEEKFL	Variable text, size 3	Último dia da semana, exemplo: se for Sábado então 'Yes', se for outro dia, então 'No'
D_LASTDAYINMONTHFL	Variable text, size 3	Último dia do mês, exemplo: se for 31 de Julho então 'Yes', se for 30 de Julho, então 'No'
D_WEEKDAYFL	Variable text, size 7	Fim de semana ou dia da semana

Tabela 9 - Estrutura da Tabela DIM_DATE

4.1.3 Tamanho da Base de Dados

Os *scale factors*, ou fatores de escala, determinam a proporção na qual os dados são carregados na base de dados. No *dataset* TPC-H, os dados utilizados para a base de dados de teste devem ser escolhidos a partir do conjunto de fatores de escala fixos definidos da seguinte forma: 1, 10, 30, 100, 300, 1000, 3000, 10000, 30000, 100000. No presente *benchmark* o tamanho da base de dados é definido com referência ao fator de escala 1, ou seja, Scale Factor (SF) = 1, havendo, portanto, aproximadamente 1 GB de dados de espaço de disco. Na tabela 10 está representado o tamanho das tabelas da base de dados de teste, tanto em termos de cardinalidade, como de tamanho em MB.

Nome da Tabela	Cardinalidade (em linhas)	Tamanho da Tabela (em MB)
SUPPLIER	10.000	2
PART	200.000	30
PARTSUPP	800.000	110
CUSTOMER	150.000	26
ORDERS	1.500.000	149
LINEITEM	6.001.215	641
NATION	25	< 1
REGION	5	< 1
Total	8661.245 linhas	956 MB

Tabela 10 – Tamanho da Base de Dados TPC-H

Devido às modificações identificadas no capítulo anterior, do modelo relacional do TPC-H para o do SSB, a cardinalidade e tamanho das tabelas sofreu também alterações. Na tabela 11 estão discriminados os valores finais das mesmas:

Nome da Tabela	Cardinalidade (em linhas)	Tamanho da Tabela (em MB)
DIM_SUPPLIER	90.000	8
DIM_PART	8.000.000	899
DIM_CUSTOMER	1.400.000	138
DIM_DATE	2.200	0,136
FACT_LINEORDER	24.000.000	2820
Total	334.922.00 linhas	3865,136 MB = 3,86 GB

Tabela 11 - Tamanho da Base de Dados SSB

4.2 Configuração dos Cloud Data Warehouses

De modo a obter as configurações das tecnologias o mais similares possível e também a tornar o *benchmark* o mais exato possível, foram selecionadas em todas as tecnologias as respectivas configurações que utilizassem dois *cores*. A única exceção é o Google BigQuery, visto esta ser uma tecnologia *serverless* e não haver uma configuração em que pudessem ser selecionadas a utilização de dois *cores*. A empresa na qual a presente dissertação está associada teve conhecimento desta discrepância, mas considerou de qualquer forma relevante a comparação de métricas também com esta tecnologia. Na tabela 12 encontram-se descritas as configurações selecionadas para cada uma das tecnologias utilizadas no presente estudo.

Tecnologia	Configurações
Snowflake	XS (com 2 cores)
Google BigQuery	Segue a abordagem <i>serverless</i>
Amazon Redshift	dc2.large (com 2 cores)
Azure Synapse	DW1000c (com 2 <i>computer nodes</i>)

Tabela 12 - Configuração das Tecnologias de CDW

4.3 Construção do Ambiente de Cloud Data Warehousing

Após a seleção do *dataset* a ser utilizado e a configuração devida dos ambientes de CDW, é possível iniciar a construção dos mesmos nas diferentes tecnologias. Para tal, foram criados dois *schemas*, a Staging Area (SA) e a Support Application Data Lake (SADL). O primeiro *schema* contém o *dataset* do TPC-H, especificado nos capítulos anteriores, pois este contém informação *raw*, sem alterações ou cruzamento de tabelas. O SADL tem um modelo que segue a 3rd Normal Form, removendo redundâncias, duplicados, criando cruzamentos de várias tabelas numa só, etc. Assim, este *schema* já segue o modelo de dados do conhecido SSB, que tem por base o *dataset* do TPC-H Benchmark, mas contém melhorias que implementam um esquema em estrela puro tradicional.

No presente capítulo é demonstrada a criação dos *schemas* mencionados, e como os respetivos dados foram carregados, de modo a oferecer uma clara e completa compreensão

sobre toda a construção de um ambiente de CDW. Neste capítulo será demonstrado detalhadamente como realizar este processo, usando como exemplo base a tecnologia Snowflake, visto este processo ser bastante similar nas outras e, portanto, essa informação se encontrar em Anexos.

4.3.1 Construção da Base de Dados e da Staging Area

O primeiro passo para a construção dos CDW foi a criação da base de dados e do *schema* SA. A SA é uma área de armazenamento temporário entre as fontes de dados e um DW, utilizada principalmente para extrair dados rapidamente das suas fontes de dados, minimizando o impacto das fontes.

No Snowflake, o processo de criação é bastante intuitivo, sendo apenas necessário acessar à área “Data” / “Databases” e especificar a denominação desejada para a base de dados e o schema da SA. Os passos são demonstrados na figura 6. Para as restantes tecnologias, este processo pode ser consultado no Anexo B – Construção da Base de Dados e da Staging Area.

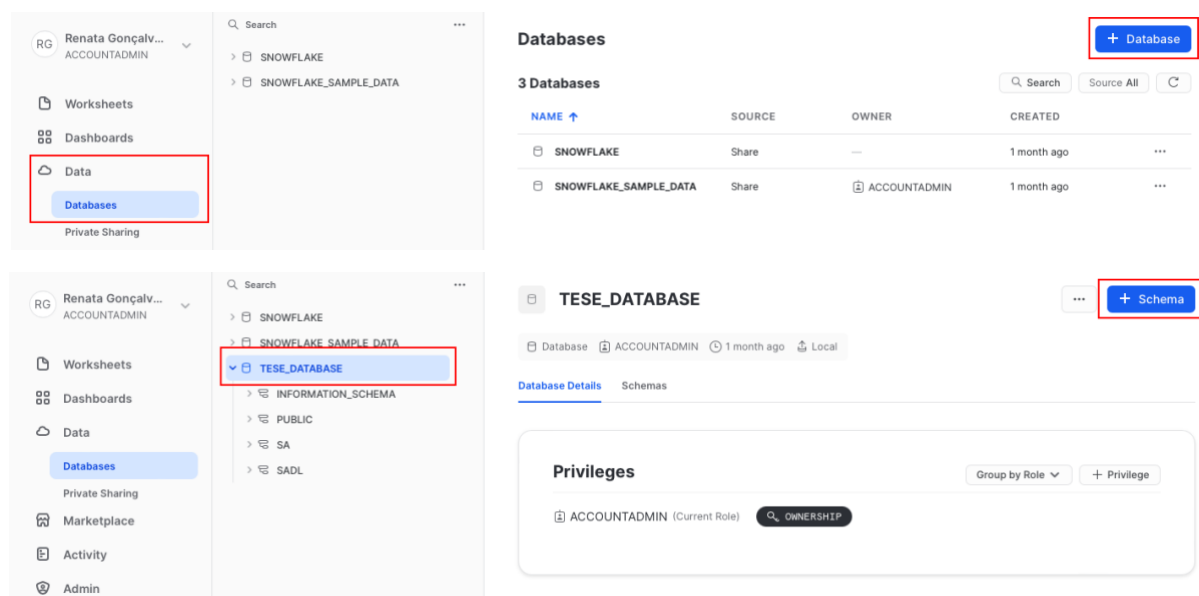


Figura 6 - Criação da Base de Dados e SA no Snowflake

4.3.2 Carregamento de Dados na Staging Area

Para o carregamento de dados para o *schema* da SA foi seguido o *dataset* do TPC-H, como explicado no capítulo anterior, de modo a, posteriormente, este poder evoluir para o

esquema do SSB. Para tal, iniciou-se pela criação das tabelas CUSTOMER, SUPPLIER, PART, PARTSUPP, LINEITEM, ORDERS, NATION e REGION. Para a construção destas tabelas, foi utilizada como base a informação presente no Anexo A – Estrutura das Tabelas do Schema TPC-H, onde é especificado o nome da tabela, das colunas e o respetivo tipo de dados presente em cada uma. Na figura 7 encontra-se um exemplo do Data Definition Language (DDL), ou linguagem de definição de dados, utilizado para a criação da tabela CUSTOMER, em que “TESE_DATABASE” representa o nome da base de dados, “SA” representa o nome do *schema* e “CUSTOMER” representa o nome da tabela. Os DDLs das restantes tabelas estão presentes no Apêndice A – Criação das Tabelas na SA, visto estes, e qualquer outro código SQL utilizado no decorrer da presente dissertação, terem autoria própria.

```
CREATE OR REPLACE TABLE TESE_DATABASE.SA.CUSTOMER (  
  c_custkey      INTEGER NOT NULL,  
  c_name         VARCHAR(25) NOT NULL,  
  c_address      VARCHAR(40) NOT NULL,  
  c_nationkey    INTEGER NOT NULL,  
  c_phone        CHAR(15) NOT NULL,  
  c_acctbal      DECIMAL(15,2) NOT NULL,  
  c_mktsegment   CHAR(10) NOT NULL,  
  c_comment      VARCHAR(117) NOT NULL,  
  PRIMARY KEY (c_custkey),  
  FOREIGN KEY (c_nationkey) REFERENCES nation (n_nationkey));
```

Figura 7 - DDL da Tabela CUSTOMER na SA

Após a criação de todas as tabelas, procedeu-se ao carregamento dos dados do *dataset* TPC-H. O Snowflake e o Redshift têm, por defeito, uma base de dados denominada SNOWFLAKE_SAMPLE_DATA e sample_data_dev, onde existem os dados do *dataset* do TPC-H, como dados de exemplo. No caso destas tecnologias, foi apenas necessário executar o comando INSERT INTO, de forma a mover os dados das suas base de dados exemplo para a base de dados criada para a presente dissertação, TESE_DATABASE. Relativamente ao BigQuery e Azure Synapse, nestas tecnologias não foi necessária a criação das várias tabelas através de código SQL, visto que estas apresentam uma funcionalidade de carregamento de ficheiros CSV para os seus ambientes, criando automaticamente o esquema das tabelas. Este processo pode ser consultado no Anexo C – Carregamento de Dados na Staging Area.

4.3.3 Construção e Carregamento de dados no Schema SADL

O objetivo da criação do *schema* SADL é a construção do esquema em estrela previamente especificado no capítulo 4.1, e que segue o modelo de dados do conhecido SSB. Como já foi explicitado, este modelo tem por base o *dataset* do TPC-H Benchmark, ou seja, a

estrutura de dados previamente criada e carregada para o *schema* SA, mas contém alterações que implementam um esquema em estrela puro tradicional.

Primeiramente, procedeu-se à criação do *schema* SADL no ambiente de todas as tecnologias, sendo que, para tal, foi apenas necessário repetir os passos efetuados para a criação do *schema* SA. De seguida, procedeu-se à criação das tabelas de dimensões e de factos especificadas no capítulo 4.1.2, especificamente DIM_CUSTOMER, DIM_SUPPLIER, DIM_PART, DIM_DATE e FACT_LINEORDER.

O principal objetivo é o carregamento dos dados nas respetivas tabelas de dimensões e factos, consoante as alterações especificadas no capítulo 4.1.2. Para tal, após a criação de novas tabelas consoante o SSB *schema*, procedeu-se à criação de *stored procedures*, um para cada uma das tabelas, sendo estes pedaços preparados de código SQL que se pode salvar para que o código possa ser reutilizado repetidamente. Nestes *stored procedures* são então realizadas todas as alterações necessárias para criar o esquema em estrela pretendido, como por exemplo, na figura 8, é possível ver código da DIM_CUSTOMER, onde se transforma o C_CITY, sendo adicionados os números de 1 a 9 no final dos dados já existentes no N_NAME da tabela NATION do *schema* SA.

```
CREATE OR REPLACE PROCEDURE TESE_DATABASE.SADL."SADL_DIM_CUSTOMER_LOAD" ()
RETURNS VARCHAR(16777216)
LANGUAGE JAVASCRIPT
EXECUTE AS OWNER
AS '
    var query_text = `

INSERT INTO TESE_DATABASE.SADL.DIM_CUSTOMER (
WITH COUNTER_9(V) AS (
    SELECT 1 FROM DUAL
    UNION ALL
    SELECT V + 1 FROM COUNTER_9
    WHERE V < 9
),
JOIN_DATA AS (
    SELECT *
    FROM TESE_DATABASE.SA.CUSTOMER
    JOIN COUNTER_9
)
SELECT
    JOIN_DATA.C_CUSTKEY,
    JOIN_DATA.C_NAME,
    JOIN_DATA.C_ADDRESS,
    CONCAT(LEFT(NATION.N_NAME, 9), JOIN_DATA.V) AS C_CITY,
    NATION.N_NAME AS C_NATION,
    REGION.R_NAME AS C_REGION,
    JOIN_DATA.C_PHONE,
    JOIN_DATA.C_MKTSEGMENT
FROM JOIN_DATA
LEFT JOIN TESE_DATABASE.SA.NATION NATION
ON JOIN_DATA.C_NATIONKEY = NATION.N_NATIONKEY
LEFT JOIN TESE_DATABASE.SA.REGION REGION
ON NATION.N_REGIONKEY = REGION.R_REGIONKEY);

`;

var statement = snowflake.createStatement({
    sqlText: query_text
});

statement.execute();
';
```

Figura 8 – Stored Procedure SADL_DIM_CUSTOMER_LOAD

Todos os *stored procedures*, apesar de muito similares, são diferentes em cada uma das tecnologias, pois algumas funções não estão presentes em todas, ou usam um diferente esquema de código para poderem ser executadas. Portanto, esta etapa do processo de construção dos CDWs foi das mais complexas visto que todo o código SQL é de autoria própria e foi necessário adaptá-lo a cada uma das tecnologias. Nos Apêndices pode ser consultado o código executado em cada uma das tecnologias, tanto para a criação das tabelas da SADL no Apêndice B – Criação das Tabelas na SADL, como para a criação dos *stored procedures* no Apêndice C – Stored Procedures para Carregamento de Dados na SADL.

Com as tabelas do *schema* SADL preenchidas, o ambiente está pronto para a realização dos testes definidos no âmbito desta dissertação. Estes testes, e os respetivos resultados para cada uma das tecnologias serão especificados e apresentados nos seguintes capítulos.

4.4 Definição das Métricas a Testar

O principal objetivo da presente dissertação é a avaliação do desempenho de cada uma das tecnologias de CDW através de duas métricas, respetivamente a avaliação do tempo de carregamento dos dados para as tabelas do *schema* SADL, através da execução dos *stored procedures* criados, e do tempo de execução de *queries* pré-definidas. De modo a obter os tempos mais precisos possível, em todas as tecnologias foi desativado o armazenamento de *queries* em *cache* e para todas as execuções foi utilizada a mesma ligação WiFi. Para cada tabela ou *query* foram executadas 5 vezes o mesmo teste, e realizada uma média dos valores, de modo a tornar o *benchmark* mais preciso. No presente capítulo serão especificados como a realização destes testes será efetuada, incluindo o código necessário para executar os *stored procedures* e as *queries* que foram selecionadas para a avaliação desta métrica.

4.4.1 Tempo de Carregamento das Tabelas da SADL

De modo a avaliar o tempo de carregamento dos dados para as tabelas da SADL, foram realizadas CALLs dos *stored procedures*, 5 vezes cada um, e realizada uma média dos 5 valores. Para tal, basta executar a função de CALL PROCEDURE ou EXEC PROCEDURE no Synapse. Na figura 9 está presente um exemplo do carregamento da tabela DIM_CUSTOMER no Snowflake e tempo que demorou a ser executado. No BigQuery, o processo é bastante similar, no

entanto, os valores apresentados encontram-se em segundos, com 0 casas decimais, enquanto nos restantes se encontram com 1 casa decimal. De forma a tornar o *benchmark* o mais preciso possível, foi utilizado o BQ Visualizer, um website que permite visualizar as *queries* executadas e informações detalhadas sobre a sua execução, obtendo assim informações dos tempos de execução com 1 casa decimal. A interface deste pode ser consultada no Anexo D – Interface do BQ Visualizer.

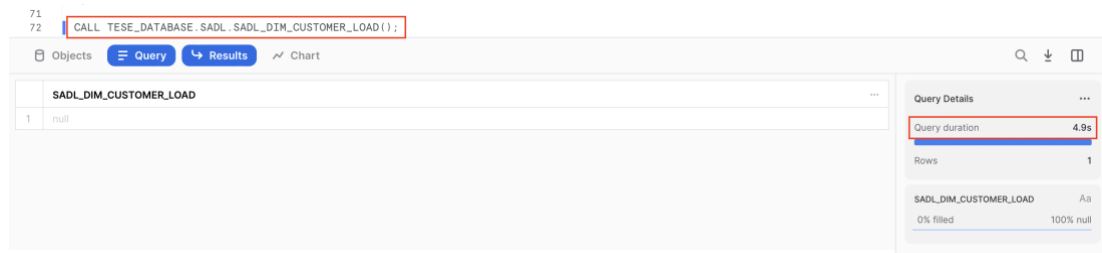


Figura 9 - Medição do Tempo de Carregamento de dados para a tabela DIM_CUSTOMER

4.4.2 Tempo de Execução de Queries

A segunda métrica selecionada foi o tempo de execução de um conjunto de *queries*, em cada uma das tecnologias, sendo estas as *queries* definidas no documento do SSB, por O'Neil et al. (2009). As *queries* de referência são escolhidas, tanto quanto possível, de modo a abranger as tarefas executadas por um importante conjunto de *queries* do *star schema*, para que os utilizadores possam obter uma classificação do desempenho do CDW daquilo que esperam usar na prática. Assim sendo, algumas das *queries* modelo são baseadas no conjunto de *queries* do TPC-H, mas foram então modificadas de modo a variar a sua seletividade. Portanto, em comparação com as 22 *queries* do TPC-H, o SSB contém quatro conjuntos de *queries*, cada um composto por três a quatro *queries* com seletividade de restrições variável, chegando a um total de 13 *queries* (Sanchez, 2016). As várias *queries* podem mais uma vez ser consultadas com maior detalhe no documento de O'Neil (2009), sendo que na figura 10 é possível analisar um exemplo de uma destas.

```

select d_year, s_nation, p_category, sum(lo_revenue - lo_supplycost) as profit
from dim_date, dim_customer, dim_supplier, dim_part, fact_lineorder
where lo_custkey = c_custkey
and lo_suppkey = s_suppkey
and lo_partkey = p_partkey
and lo_orderdate = d_datekey
and c_region = 'AMERICA'
and s_region = 'AMERICA'
and (d_year = 1997 or d_year = 1998)
and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
group by d_year, s_nation, p_category
order by d_year, s_nation, p_category;

```

Figura 10 - Exemplo de Query do SSB

Por fim, na tabela 13 estão descritos os números de linhas apresentados na execução de cada uma das queries.

Query	Número de Linhas
Query 1.1	1
Query 1.2	1
Query 1.3	1
Query 2.1	240
Query 2.2	48
Query 2.3	6
Query 3.1	150
Query 3.2	486
Query 3.3	24
Query 3.4	4
Query 4.1	30
Query 4.2	50
Query 4.3	360

Tabela 13 - Número de Linhas Retornado em cada Query

5. RESULTADOS OBTIDOS

No presente capítulo serão apresentados os resultados obtidos da avaliação das métricas definidas anteriormente, respetivamente a avaliação do tempo de carregamento dos dados para as tabelas do *schema* SADL, através da execução dos *stored procedures* criados, e do tempo de execução de *queries* pré-definidas, com o objetivo de avaliar o desempenho das quatro tecnologias. Para cada um dos testes, foram executadas as *queries* 5 vezes cada uma, sendo realizada uma média das 5 execuções, de forma a tornar o *benchmark* mais preciso. Adicionalmente, e tal como já foi referido anteriormente, em todas as tecnologias foi desativado o armazenamento em cache, de modo que os resultados não sejam afetados pelas execuções anteriores.

Na figura 11 estão apresentados os valores, em segundos, da média dos tempos de carregamento dos dados para as respetivas tabelas da SADL. O carregamento foi realizado através da execução dos *stored procedures* criados, por autoria própria, em cada uma das tecnologias. Na tabela 14 pode também ser consultada a mesma informação.

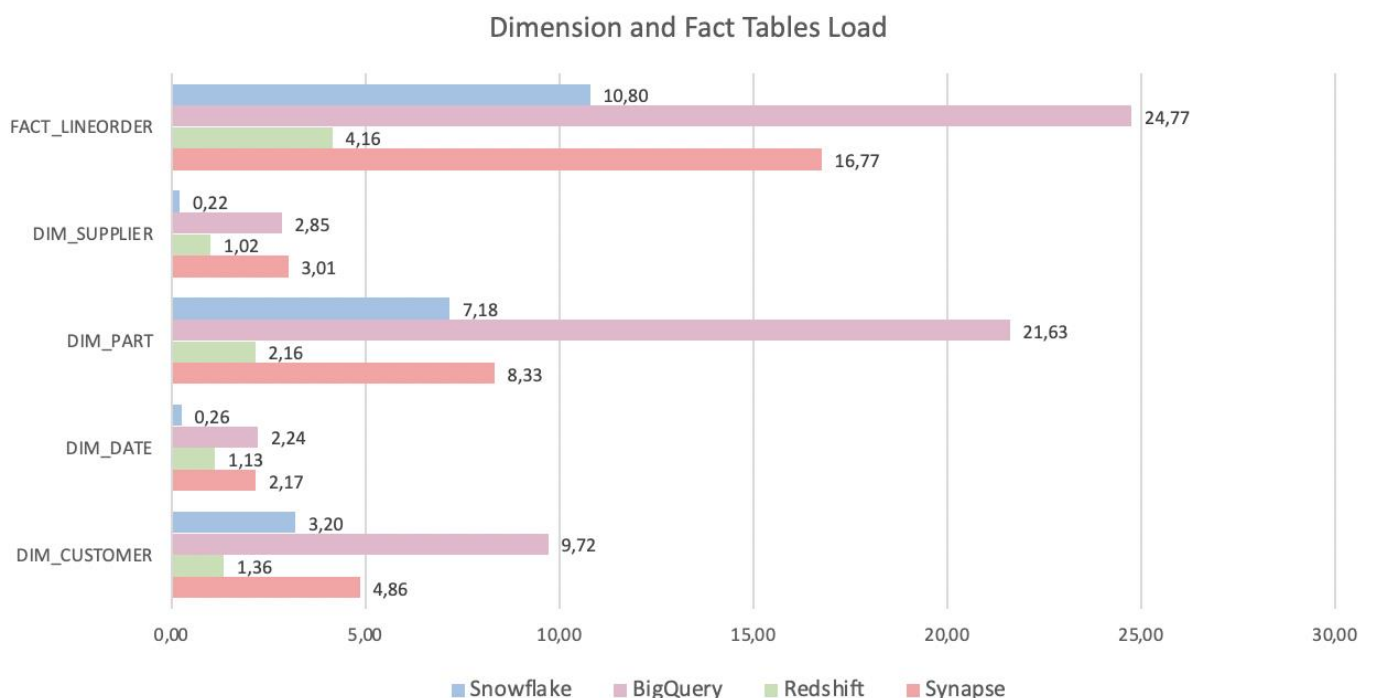


Figura 11 - Média dos tempos de carregamento dos dados nas respetivas tabelas da SADL

Nome da Tabela	Azure Synapse	Amazon Redshift	Google BigQuery	Snowflake
DIM_CUSTOMER	4,86	1,36	9,72	3,20
DIM_DATE	2,17	1,13	2,24	0,26
DIM_PART	8,33	2,16	21,63	7,18
DIM_SUPPLIER	3,01	1,02	2,85	0,22
FACT_LINEORDER	16,77	4,16	24,77	10,80

Tabela 14 - Média dos tempos de carregamento dos dados nas respectivas tabelas da SADL

Como pode ser observado, em todas as tabelas, exceto na DIM_SUPPLIER, o Google BigQuery apresentou um desempenho mais demorado, em comparação às outras tecnologias. Estes valores podem ter sido afetados por o BigQuery ser uma tecnologia *serverless*, enquanto as outras tecnologias estavam todas configuradas com 2 *cores*. Adicionalmente, a DIM_SUPPLIER é das tabela mais pequenas, com apenas 90 mil linhas e 8 MB, podendo residir aí a razão de uma ligeira maior rapidez de carregamento de dados nesse caso, em comparação ao Azure Synapse. Retirando este caso, o Synapse também apresentou um desempenho mais demorado, ficando sempre em 3º lugar em comparação às restantes. Assim, é possível concluir que as tecnologias mais competitivas, neste ambiente que foi configurado e desenvolvido, foram o Snowflake e o Amazon Redshift. O Snowflake apresentou um melhor desempenho no carregamento de dados nas tabelas DIM_SUPPLIER e DIM_DATE, que são as duas tabelas com menor volume de dados, com, respetivamente, 90 mil linhas e 8MB e 2200 linhas e 0,136MB cada uma. Assim, observa-se que, nas dadas condições, o Snowflake apresenta um melhor desempenho de carregamento de dados em tabelas com um menor volume de dados e o Amazon Redshift um melhor desempenho em tabelas com um maior volume de dados.

Relativamente à segunda métrica selecionada, a seguinte figura 12 apresenta os valores, em segundos, da média dos tempos de execução das várias *queries* definidas no documento de O'Neil (2009), sendo estas selecionadas com o objetivo de abranger as tarefas executadas por um importante conjunto de *queries* do *star schema*, para que os utilizadores possam obter uma classificação do desempenho do CDW daquilo que esperam usar na prática. Na tabela 15 pode também ser consultada a mesma informação.

Queries Run

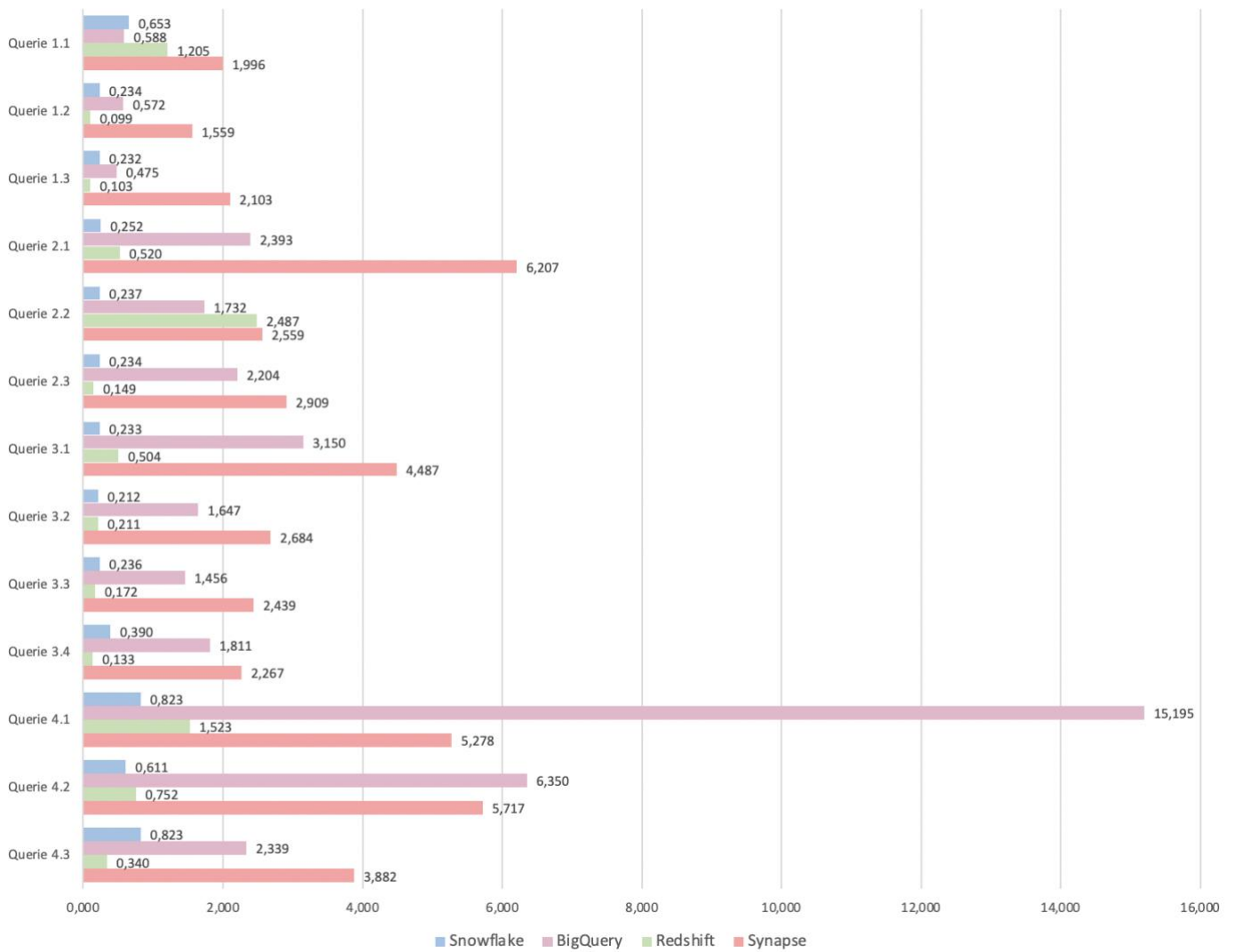


Figura 12 - Média dos tempos de execução das queries

Query	Snowflake	Google BigQuery	Amazon Redshift	Azure Synapse
Query 1.1	0,653	0,588	1,205	1,996
Query 1.2	0,234	0,572	0,099	1,559
Query 1.3	0,232	0,475	0,103	2,103
Query 2.1	0,252	2,393	0,520	6,207
Query 2.2	0,237	1,732	2,487	2,559
Query 2.3	0,234	2,204	0,149	2,909
Query 3.1	0,233	3,150	0,504	4,487
Query 3.2	0,212	1,647	0,211	2,684
Query 3.3	0,236	1,456	0,172	2,439

Query 3.4	0,390	1,811	0,133	2,267
Query 4.1	0,823	15,195	1,523	5,278
Query 4.2	0,611	6,350	0,752	5,717
Query 4.3	0,823	2,339	0,340	3,882

Tabela 15 - Média dos tempos de execução das queries

Analisando o gráfico, observa-se que o Azure Synapse apresentou um desempenho mais lento em praticamente todas as execuções de *queries*, exceto na Query 4.1 e 4.2. Este quarto conjunto de *queries* é agrupado por campos em duas dimensões e tem restrições em três dimensões, onde o objetivo é a medição do lucro agregado. Segundo O’Neil (2009), dos quatro conjuntos este é o mais complexo, podendo residir aí a razão para o Google BigQuery ter apresentado um desempenho mais demorado na maioria das *queries* do conjunto número 4. Excluindo este caso, e as execuções das *queries* 1.1 e 2.2, o Google BigQuery apresentou sempre um menor desempenho em comparação ao Snowflake e ao Amazon Redshift, ficando quase sempre colocado em 3º lugar em comparação às restantes. Na avaliação desta métrica, mais uma vez, o Snowflake e o Redshift apresentam-se como as tecnologias mais competitivas, segundo as condições configuradas e desenvolvidas. Estas tecnologias apresentam valores bastantes parecidos em quase todas as execuções, havendo uma maior discrepância nas queries 1.1, 2.2, 4.1 e 4.3, em que nas primeiras três o Snowflake teve um tempo de execução mais rápido, e na última teve o Redshift.

De forma geral, conclui-se que os resultados foram mais favoráveis no Snowflake e no Redshift, sendo então estes recomendados aos utilizadores que considerem as métricas utilizadas como uma das suas prioridades na seleção de uma tecnologia de CDW. Com esta análise, o objetivo principal do desenvolvimento da presente dissertação foi alcançado, sendo esse a avaliação do desempenho das várias tecnologias selecionadas, num determinado ambiente, e segundo um conjunto de métricas.

6. CONCLUSÃO

Através da elaboração da presente dissertação foi possível contribuir para a resolução do problema inicialmente apresentado. Este residia no facto de que selecionar a tecnologia mais adequada para um CDW ser uma tarefa complexa, pois existe um grande número de fatores que influenciam esta decisão e porque há uma grande oferta existente no mercado atual. Através da leitura da presente dissertação, um utilizador interessado na avaliação do desempenho dos CDW, segundo as métricas selecionadas, a configuração dos ambientes criados, e no tamanho dos dados processados, pode retirar conclusões interessantes e úteis para o auxiliar na decisão de qual tecnologia de CDW adotar.

Segundo os resultados obtidos e apresentados no capítulo anterior, é possível concluir que estes foram mais favoráveis no Snowflake e no Redshift, pois apresentaram um desempenho mais rápido nas duas métricas selecionadas, estando quase sempre com valores bastante aproximados uns do outro. É necessário ter em atenção que estes foram os valores obtidos segundo as configurações selecionadas e uma mudança nas mesmas pode trazer variações nos tempos medidos, o que pode ser uma variante interessante no trabalho apresentado.

Desta forma, é possível afirmar que os seguintes objetivos e resultados, discriminados no início do documento, foram alcançados:

- “Apresentação de uma revisão de literatura sobre a área de armazenamento de dados na *cloud*, discutindo os seus principais conceitos e estudando as gerações existentes deste tipo de tecnologias”, onde foi possível aprender e ganhar uma base de conhecimento sobre o tema, o que permitiu realizar um trabalho mais informado para a presente dissertação;
- “Apresentação de um conjunto de tecnologias de CDW com maior presença no mercado atual e de um estudo de mercado de algumas previamente selecionadas, relativamente a características como arquitetura, escalabilidade, segurança e conformidade, suporte de dados e preços”. O resultado deste tópico encontra-se apresentado no capítulo 3.2 da presente dissertação onde o objetivo era o estudo do conjunto de características mencionadas com o intuito de obter um maior conhecimento base sobre as várias tecnologias. No entanto, este estudo não

influenciou os resultados obtidos no final da dissertação, sendo que apenas a característica de desempenho de cada um dos CDW foi avaliada e foram retiradas conclusões sobre a mesma;

- “Construção de um CDW em quatro tecnologias de CDW, utilizando um *dataset* pré-definido”, onde foi possível criar uma familiarização com as tecnologias selecionadas e desenvolver o ambiente base para a realização do objetivo final da dissertação, discriminado no seguinte tópico; e
- “Partilha dos resultados de um benchmark de quatro tecnologias de CDW, utilizando um conjunto de métricas, com o objetivo de obter resultados relativos ao desempenho de cada uma”, onde aqui foram obtidos os resultados que contribuíram para a questão inicial desde projeto, sendo ela a decisão de qual tecnologia de CDW utilizar.

Por fim, é considerado que a presente dissertação pode dar aso a trabalhos e investigações futuras, como por exemplo, comparação dos resultados apresentados a resultados em plataformas on-premises, ou comparação dos resultados aos que seriam obtidos se as configurações dos ambientes fossem diferentes, por exemplo, mais que 2 cores, ou ainda comparação com mais tecnologias de CDW, entre outras possíveis variações. Este estudo adicional pode ser bastante interessante de modo a tornar o *benchmark* mais completo e com uma maior utilidade para o utilizador final. Uma outra variação poderia ser a aposta num estudo mais aprofundado de características dos CDW para além do seu desempenho, que foi efetivamente o foco nesta dissertação. Outros exemplos de características passam pela arquitetura, escalabilidade, segurança e conformidade, suporte de dados e preços, que foram referidos na presente dissertação mas não foram fatores influenciadores na decisão final de escolha de tecnologia de CDW.

REFERÊNCIAS

- Ahmad Dar, A. (2018). Cloud computing-positive impacts and challenges in business perspective. *Journal of Computer Science & Systems Biology*, 12(01). doi: <https://doi.org/10.4172/jcsb.1000294>
- Alkhalil, A., Sahandi, R., & John, D. (2017). An exploration of the determinants for decision to migrate existing resources to cloud computing using an integrated toe-DOI model. *Journal of Cloud Computing*, 6(1). doi: <https://doi.org/10.1186/s13677-016-0072-x>
- Awoyelu, I., Omodunbi, T., & Udo, J. (2013). Bridging the gap in modern computing infrastructures: Issues and challenges of data warehousing and cloud computing. *Computer and Information Science*, 7(1). doi:10.5539/cis.v7n1p33
- Cloud Industry Forum. (2020). 8 criteria to ensure you select the Right Cloud Service Provider. 8 criteria to ensure you select the right cloud service provider. Retrieved from <https://www.cloudindustryforum.org/content/8-criteria-ensure-you-select-right-cloud-service-provider>
- Geerts, G. L. (2011). A design science research methodology and its application to Accounting Information Systems Research. *International Journal of Accounting Information Systems*, 12(2), 142-151. doi:10.1016/j.accinf.2011.02.004
- Gill, S. S., Tuli, S., Xu, M., Singh, I., Singh, K. V., Lindsay, D., ... Garraghan, P. (2019). Transformative effects of IOT, blockchain and Artificial Intelligence on Cloud Computing: Evolution, vision, trends and open challenges. *Internet of Things*, 8, 100118. doi: <https://doi.org/10.1016/j.iot.2019.100118>
- Guermazi, E., Ayed, M. B., & Ben-Abdallah, H. (2015). Adaptive Security for Cloud Data Warehouse as a Service. Retrieved from <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7166672>
- Haris, M., & Khan, R. Z. (2018). A systematic review on cloud computing. *International Journal of Computer Sciences and Engineering*, 6(11), 632-639. doi: <https://doi.org/10.26438/ijcse/v6i11.632639>
- IBM Cloud Education (2021). IaaS vs. PaaS v. SaaS. Retrieved from <https://www.ibm.com/cloud/learn/iaas-paas-saas>
- Indicative Team. (2021). What is a TPC-H benchmark? Retrieved from <https://www.indicative.com/resource/tcp-h-benchmark/>

- Kimball, R., & Ross, M. (2002). *The data warehouse toolkit: the complete guide to dimensional modeling*. John Wiley & Sons. Retrieved from https://books.google.pt/books?hl=pt-PT&lr=&id=XoS2oy1IcB4C&oi=fnd&pg=PA1&dq=The+Data+Warehouse+Toolkit+kimball&ots=1DIcnGgLcH&sig=NIqQVfywsNrkYNMTEb6DSVltybs&redir_esc=y#v=onepage&q=The%20Data%20Warehouse%20Toolkit%20kimball&f=false
- King, T. (2021). The 10 best cloud data warehouse solutions to consider in 2022. *Best Data Management Software, Vendors and Data Science Platforms*. Retrieved July 28, 2022, from <https://solutionsreview.com/data-management/the-best-cloud-data-warehouse-solutions-2/>
- Kline, L. (2021). *Azure Synapse vs snowflake: The definitive guide*. Retrieved from <https://hightouch.io/blog/azure-synapse-vs-snowflake-the-definitive-guide/>
- Kline, L. (2021). *Bigquery vs snowflake: The definitive guide*. Retrieved from <https://hightouch.io/blog/big-query-vs-snowflake-the-definitive-guide/>
- Kline, L. (2022). *Redshift vs snowflake: The definitive guide*. Retrieved from <https://hightouch.io/blog/redshift-vs-snowflake-the-definitive-guide/>
- Miloslavskaya, N., & Tolstoy, A. (2016). Application of big data, fast data, and Data Lake Concepts to information security issues. *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. doi:10.1109/w-ficloud.2016.41
- Moody, D. L., & Kortink, M. A. (2000). From enterprise models to dimensional models: a methodology for data warehouse and data mart design. Retrieved from https://www.researchgate.net/publication/220841952_From_enterprise_models_to_dimensional_models_a_methodology_for_data_warehouse_and_data_mart_design
- Nieuwenhuis, L. J., Ehrenhard, M. L., & Prause, L. (2018). The shift to cloud computing: The impact of disruptive technology on the enterprise software business ecosystem. *Technological Forecasting and Social Change*, 129, 308-313. doi: <https://doi.org/10.1016/j.techfore.2017.09.037>
- O'Neil, P., O'Neil, B., & Chen, X. (2009). *Star schema benchmark*. Retrieved from <https://www.cs.umb.edu/~poneil/StarSchemaB.PDF>
- Peppers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3), 45-77. doi:10.2753/mis0742-1222240302
- Rehman, K. U., Ahmad, U., & Mahmood, S. (2018). *A comparative analysis of traditional and Cloud Data Warehouse*. Retrieved from https://www.researchgate.net/profile/Sajid-Mahmood-11/publication/329704023_A_Comparative_Analysis_of_Traditional_and_Cloud_Data

[_Warehouse/links/5c3c638c92851c22a3736ebb/A-Comparative-Analysis-of-Traditional-and-Cloud-Data-Warehouse.pdf](#)

Sanchez, J. (2016). A review of STAR schema benchmark. Retrieved from <https://arxiv.org/pdf/1606.00295.pdf>

Sanchez, J. C. (2016). INVESTIGATING THE STAR SCHEMA BENCHMARK AS A REPLACEMENT FOR THE TPC-H DECISION SUPPORT SYSTEM. Retrieved from <https://thescholarship.ecu.edu/bitstream/handle/10342/6036/SANCHEZ-MASTERSTHESIS-2016.pdf;sequence=1>

SAP. (2020). What is a data warehouse?: Definition, components, architecture: SAP insights. SAP. Retrieved from <https://www.sap.com/insights/what-is-a-data-warehouse.html>

Singhal, U. (2021). Competitions, quizzes, hackathons, scholarships and internships for students and corporates. Dare2Compete. Retrieved from <https://dare2compete.com/blog/characteristics-of-data-warehouse>

Sunyaev, A. (2020). Cloud computing. *Internet Computing*, 195-236. doi: https://doi.org/10.1007/978-3-030-34957-8_7

Thompson, W. J., & Van der Walt, J. S. (2010). Business intelligence in the cloud. *SA Journal of Information Management*, 12(1). doi:10.4102/sajim.v12i1.445

Virant, R., Kamišalić, A., & Šestak, M. (2021). A comparison of traditional and modern data warehouse architectures. Retrieved from <https://dk.um.si/IzpisGradiva.php?id=80484>

Wurm, A. H. (2022). Data Warehouse Technology Value Matrix 2022 - oracle.com. Retrieved from <https://www.oracle.com/a/ocom/docs/database/nucleus-research-data-warehouse-technology-matrix-report.pdf>

APÊNDICES

Apêndice A – Criação das Tabelas na SA

De seguida, encontra-se apresentado o DDL para a criação das tabelas conforme o schema do TPC-H, ou seja, das tabelas presentes no *schema* SA, sendo este o mesmo código em todas as tecnologias.

```
CREATE OR REPLACE TABLE region ( r_regionkey INTEGER NOT NULL,  
    r_name          CHAR(25) NOT NULL,  
    r_comment      VARCHAR(152),  
    PRIMARY KEY (r_regionkey));
```

```
CREATE OR REPLACE TABLE nation ( n_nationkey INTEGER NOT NULL,  
    n_name          CHAR(25) NOT NULL,  
    n_regionkey    INTEGER NOT NULL,  
    n_comment      VARCHAR(152),  
    PRIMARY KEY (n_nationkey),  
    FOREIGN KEY (n_regionkey) REFERENCES region  
(r_regionkey));
```

```
CREATE OR REPLACE TABLE part ( p_partkey      INTEGER NOT NULL,  
    p_name         VARCHAR(55) NOT NULL,  
    p_mfgr         CHAR(25) NOT NULL,  
    p_brand        CHAR(10) NOT NULL,  
    p_type         VARCHAR(25) NOT NULL,  
    p_size         INTEGER NOT NULL,  
    p_container    CHAR(10) NOT NULL,  
    p_retailprice  DECIMAL(15,2) NOT NULL,  
    p_comment      VARCHAR(23) NOT NULL,  
    PRIMARY KEY (p_partkey));
```

```
CREATE OR REPLACE TABLE supplier ( s_suppkey      INTEGER NOT NULL,  
    s_name         CHAR(25) NOT NULL,  
    s_address      VARCHAR(40) NOT NULL,  
    s_nationkey    INTEGER NOT NULL,  
    s_phone        CHAR(15) NOT NULL,  
    s_acctbal      DECIMAL(15,2) NOT NULL,  
    s_comment      VARCHAR(101) NOT NULL,  
    PRIMARY KEY (s_suppkey),  
    FOREIGN KEY (s_nationkey) REFERENCES nation  
(n_nationkey));
```

```
CREATE OR REPLACE TABLE partsupp ( ps_partkey      INTEGER NOT NULL,  
    ps_suppkey     INTEGER NOT NULL,  
    ps_availqty    INTEGER NOT NULL,  
    ps_supplycost  DECIMAL(15,2) NOT NULL,  
    ps_comment     VARCHAR(199) NOT NULL,  
    PRIMARY KEY (ps_partkey, ps_suppkey),
```

```

FOREIGN KEY (ps_partkey) REFERENCES part
(p_partkey),
FOREIGN KEY (ps_suppkey) REFERENCES supplier
(s_suppkey));

CREATE OR REPLACE TABLE customer ( c_custkey      INTEGER NOT NULL,
c_name      VARCHAR(25) NOT NULL,
c_address   VARCHAR(40) NOT NULL,
c_nationkey INTEGER NOT NULL,
c_phone     CHAR(15) NOT NULL,
c_acctbal   DECIMAL(15,2) NOT NULL,
c_mktsegment CHAR(10) NOT NULL,
c_comment   VARCHAR(117) NOT NULL,
PRIMARY KEY (c_custkey),
FOREIGN KEY (c_nationkey) REFERENCES nation
(n_nationkey));

CREATE OR REPLACE TABLE orders ( o_orderkey      INTEGER NOT NULL,
o_custkey   INTEGER NOT NULL,
o_orderstatus CHAR(1) NOT NULL,
o_totalprice DECIMAL(15,2) NOT NULL,
o_orderdate DATE NOT NULL,
o_orderpriority CHAR(15) NOT NULL,
o_clerk     CHAR(15) NOT NULL,
o_shippriority INTEGER NOT NULL,
o_comment   VARCHAR(79) NOT NULL,
PRIMARY KEY (o_orderkey),
FOREIGN KEY (o_custkey) REFERENCES customer
(c_custkey));

CREATE OR REPLACE TABLE lineitem ( l_orderkey      INTEGER NOT NULL,
l_partkey   INTEGER NOT NULL,
l_suppkey   INTEGER NOT NULL,
l_linenumber INTEGER NOT NULL,
l_quantity  DECIMAL(15,2) NOT NULL,
l_extendedprice DECIMAL(15,2) NOT NULL,
l_discount  DECIMAL(15,2) NOT NULL,
l_tax       DECIMAL(15,2) NOT NULL,
l_returnflag CHAR(1) NOT NULL,
l_linestatus CHAR(1) NOT NULL,
l_shipdate  DATE NOT NULL,
l_commitdate DATE NOT NULL,
l_receiptdate DATE NOT NULL,
l_shipinstruct CHAR(25) NOT NULL,
l_shipmode   CHAR(10) NOT NULL,
l_comment   VARCHAR(44) NOT NULL,
PRIMARY KEY (l_orderkey, l_linenumber),
FOREIGN KEY (l_orderkey) REFERENCES orders
(o_orderkey),
FOREIGN KEY (l_partkey, l_suppkey) REFERENCES
partsupp (ps_partkey, ps_suppkey));

```

Apêndice B – Criação das Tabelas na SADL

De seguida, encontra-se apresentado o DDL para a criação das tabelas conforme o schema do SSB, ou seja, das tabelas presentes no *schema* SADL, sendo este o mesmo código em todas as tecnologias.

```
CREATE OR REPLACE TABLE tese_database.sadl.DIM_CUSTOMER (  
  c_custkey      INTEGER NOT NULL,  
  c_name        VARCHAR(25) NOT NULL,  
  c_address     VARCHAR(50) NOT NULL,  
  c_city       VARCHAR(10) NOT NULL,  
  c_nation     VARCHAR(15) NOT NULL,  
  c_region     VARCHAR(12) NOT NULL,  
  c_phone      VARCHAR(15) NOT NULL,  
  c_mktsegment VARCHAR(10) NOT NULL  
);
```

```
CREATE OR REPLACE TABLE tese_database.sadl.DIM_PART (  
  p_partkey     INTEGER NOT NULL,  
  p_name       VARCHAR(22) NOT NULL,  
  p_mfgr      VARCHAR(6) NOT NULL,  
  p_category   VARCHAR(7) NOT NULL,  
  p_brand1    VARCHAR(9) NOT NULL,  
  p_color     VARCHAR(60) NOT NULL,  
  p_type      VARCHAR(25) NOT NULL,  
  p_size      INTEGER NOT NULL,  
  p_container VARCHAR(10) NOT NULL  
);
```

```
CREATE OR REPLACE TABLE tese_database.sadl.DIM_SUPPLIER (  
  s_suppkey     INTEGER NOT NULL,  
  s_name       VARCHAR(25) NOT NULL,  
  s_address    VARCHAR(50) NOT NULL,  
  s_city      VARCHAR(10) NOT NULL,  
  s_nation    VARCHAR(15) NOT NULL,  
  s_region    VARCHAR(12) NOT NULL,  
  s_phone     VARCHAR(15) NOT NULL  
);
```

```
CREATE OR REPLACE TABLE tese_database.sadl.DIM_DATE (  
  d_datekey     DATE NOT NULL,  
  d_date       VARCHAR(10) NOT NULL,  
  d_dayofweek   VARCHAR(10) NOT NULL,  
  d_month      VARCHAR(9) NOT NULL,  
  d_year       INTEGER NOT NULL,  
  d_yearmonthnum INTEGER NOT NULL,  
  d_yearmonth  VARCHAR(7) NOT NULL,  
  d_daynuminweek INTEGER NOT NULL,  
  d_daynuminmonth INTEGER NOT NULL,  
  d_daynuminyear INTEGER NOT NULL,  
  d_monthnuminyear INTEGER NOT NULL,  
  d_weeknuminyear INTEGER NOT NULL,
```



```

d_lastdayinweekfl  VARCHAR(3) NOT NULL,
d_lastdayinmotnhfl VARCHAR(3) NOT NULL,
d_weekdayfl       VARCHAR(7) NOT NULL
);

CREATE OR REPLACE TABLE tese_database.sadl.FACT_LINEORDER (
  lo_orderkey      BIGINT NOT NULL,
  lo_linenumbers   INTEGER NOT NULL,
  lo_custkey       INTEGER NOT NULL,
  lo_partkey       INTEGER NOT NULL,
  lo_suppkey       INTEGER NOT NULL,
  lo_orderdate     DATE NOT NULL,
  lo_orderpriority VARCHAR(15) NOT NULL,
  lo_shippriority  VARCHAR(1) NOT NULL,
  lo_quantity      INTEGER NOT NULL,
  lo_extendedprice DECIMAL(20,2) NOT NULL,
  lo_ordertotalprice INTEGER NOT NULL,
  lo_discount      DECIMAL(5,2) NOT NULL,
  lo_revenue       DECIMAL(15,9) NOT NULL,
  lo_supplycost    DECIMAL(15,5) NOT NULL,
  lo_tax           INTEGER NOT NULL,
  lo_commitdate    DATE NOT NULL,
  lo_shipmode      VARCHAR(10) NOT NULL
);

```

Apêndice C – Stored Procedures para Carregamento de Dados na SADL

De seguida, encontra-se apresentado o código desenvolvido para o carregamento de dados das tabelas da SADL, sendo este código parecido em todas as tecnologias, mas tendo as suas divergências devido às alterações necessárias para este funcionar em todas as tecnologias. Primeiramente, é apresentado o código utilizado na plataforma Snowflake.

```
CREATE OR REPLACE PROCEDURE TESE_DATABASE.SADL."SADL_DIM_CUSTOMER_LOAD" ()
RETURNS VARCHAR(16777216)
LANGUAGE JAVASCRIPT
EXECUTE AS OWNER
AS '
    var query_text = `
INSERT INTO TESE_DATABASE.SADL.DIM_CUSTOMER (
WITH COUNTER_9(V) AS (
    SELECT 1 FROM DUAL
    UNION ALL
    SELECT V + 1 FROM COUNTER_9
    WHERE V < 9
),
JOIN_DATA AS (
    SELECT *
    FROM TESE_DATABASE.SA.CUSTOMER
    JOIN COUNTER_9
)
SELECT DISTINCT
    JOIN_DATA.C_CUSTKEY,
    JOIN_DATA.C_NAME,
    JOIN_DATA.C_ADDRESS,
    CONCAT(LEFT(NATION.N_NAME, 9), JOIN_DATA.V) AS C_CITY,
    NATION.N_NAME AS C_NATION,
    REGION.R_NAME AS C_REGION,
    JOIN_DATA.C_PHONE,
    JOIN_DATA.C_MKTSEGMENT
FROM JOIN_DATA
LEFT JOIN TESE_DATABASE.SA.NATION NATION
    ON JOIN_DATA.C_NATIONKEY = NATION.N_NATIONKEY
LEFT JOIN TESE_DATABASE.SA.REGION REGION
    ON NATION.N_REGIONKEY = REGION.R_REGIONKEY);
`;

var statement = snowflake.createStatement({
    sqlText: query_text
});
statement.execute();
';

CREATE OR REPLACE PROCEDURE TESE_DATABASE.SADL."SADL_DIM_DATE_LOAD" ()
RETURNS VARCHAR(16777216)
LANGUAGE JAVASCRIPT
EXECUTE AS OWNER
AS '
    var query_text = `
```

```

INSERT INTO TESE_DATABASE.SADL.DIM_DATE (
  WITH CTE_MY_DATE AS (
    SELECT DATEADD(DAY, SEQ4(), '1992-01-01') AS MY_DATE
    FROM TABLE (GENERATOR (ROWCOUNT=>2192))
  )
  SELECT
    TO_CHAR(MY_DATE, 'YYYY-MM-DD') AS d_datekey,
    TO_CHAR(MY_DATE, 'YYYY-MM-DD') AS d_date,
    DECODE (DAYNAME (MY_DATE),
      'Mon', 'Monday', 'Tue', 'Tuesday',
      'Wed', 'Wednesday', 'Thu', 'Thursday',
      'Fri', 'Friday', 'Sat', 'Saturday',
      'Sun', 'Sunday') AS d_dayofweek,
    TO_CHAR(MY_DATE, 'MMMM') AS d_month,
    YEAR(MY_DATE) AS d_year,
    YEAR(MY_DATE) || LPAD (MONTH (MY_DATE), 2, '0') AS d_yearmonthnum,
    TO_CHAR(MY_DATE, 'MONYYYY') AS d_yearmonth,
    DAYOFWEEK(MY_DATE) + 1 AS d_daynuminweek,
    DAYOFMONTH(MY_DATE) AS d_daynuminmonth,
    DAYOFYEAR(MY_DATE) AS d_daynuminyear,
    MONTH(MY_DATE) AS d_monthnuminyear,
    CASE
      WHEN DAYOFWEEK(MY_DATE) = '0' THEN WEEKOFYEAR(MY_DATE) +
1
      ELSE WEEKOFYEAR(MY_DATE)
    END AS d_weeknuminyear,
    CASE
      WHEN d_daynuminweek = 7 THEN 'Yes'
      ELSE 'No'
    END AS d_lastdayinweekfl,
    CASE
      WHEN d_daynuminmonth = '31' THEN 'Yes'
      WHEN (d_monthnuminyear = 4 OR d_monthnuminyear = 6 OR
d_monthnuminyear = 9 OR d_monthnuminyear = 11)
      AND d_daynuminmonth = '30' THEN 'Yes'
      WHEN (d_year = '1992' OR d_year = '1996') AND
d_monthnuminyear = 2 AND d_daynuminmonth = '29' THEN 'Yes'
      WHEN (d_year = '1993' OR d_year = '1994' OR d_year =
'1995' OR d_year = '1997')
      AND (d_monthnuminyear = 2) AND d_daynuminmonth = '28'
THEN 'Yes'
      ELSE 'No'
    END AS d_lastdayinmonthfl,
    CASE
      WHEN d_dayofweek = 'Saturday' OR d_dayofweek = 'Sunday'
THEN 'Weekend'
      ELSE 'Weekday'
    END AS d_weekdayfl
  FROM CTE_MY_DATE
);
';
var statement = snowflake.createStatement({
  sqlText: query_text
});
statement.execute();
';

```

```

CREATE OR REPLACE PROCEDURE TESE_DATABASE.SADL."SADL_DIM_PART_LOAD" ()
RETURNS VARCHAR(16777216)
LANGUAGE JAVASCRIPT
EXECUTE AS OWNER
AS '
    var query_text = `
INSERT INTO TESE_DATABASE.SADL.DIM_PART (
WITH COUNTER_40(V) AS (
    SELECT 1 FROM DUAL
    UNION ALL
    SELECT V + 1 FROM COUNTER_40
    WHERE V < 40
),
JOIN_DATA AS (
    SELECT *
    FROM TESE_DATABASE.SA.PART
    JOIN COUNTER_40
    ORDER BY P_PARTKEY
)
SELECT
    JOIN_DATA.P_PARTKEY,
    SUBSTRING(JOIN_DATA.P_NAME, 0, CHARINDEX(' ', JOIN_DATA.P_NAME,
CHARINDEX(' ', JOIN_DATA.P_NAME, 0)+1)) AS P_NAME,
    REPLACE(JOIN_DATA.P_MFGR,'Manufacturer', 'MFGR') AS P_MFGR,
    REPLACE(JOIN_DATA.P_BRAND,'Brand', 'MFGR') AS P_CATEGORY,
    REPLACE(CONCAT(JOIN_DATA.P_BRAND, JOIN_DATA.V),'Brand', 'MFGR') AS
P_BRAND1,
    JOIN_DATA.P_NAME AS P_COLOR,
    JOIN_DATA.P_TYPE,
    JOIN_DATA.P_SIZE,
    JOIN_DATA.P_CONTAINER
FROM JOIN_DATA
ORDER BY P_PARTKEY, P_BRAND1);
`;
var statement = snowflake.createStatement({
    sqlText: query_text
});
statement.execute();
';

```

```

CREATE OR REPLACE PROCEDURE TESE_DATABASE.SADL."SADL_DIM_SUPPLIER_LOAD" ()
RETURNS VARCHAR(16777216)
LANGUAGE JAVASCRIPT
EXECUTE AS OWNER
AS '
    var query_text = `
INSERT INTO TESE_DATABASE.SADL.DIM_SUPPLIER (
WITH COUNTER_9(V) AS (
    SELECT 1 FROM DUAL
    UNION ALL
    SELECT V + 1 FROM COUNTER_9
    WHERE V < 9
),
JOIN_DATA AS (
    SELECT *

```

```

FROM TESE_DATABASE.SA.SUPPLIER
JOIN COUNTER_9
)
SELECT DISTINCT
JOIN_DATA.S_SUPPKEY,
JOIN_DATA.S_NAME,
JOIN_DATA.S_ADDRESS,
CONCAT(LEFT(NATION.N_NAME, 9), JOIN_DATA.V) AS S_CITY,
NATION.N_NAME AS S_NATION,
REGION.R_NAME AS S_REGION,
JOIN_DATA.S_PHONE
FROM JOIN_DATA
LEFT JOIN TESE_DATABASE.SA.NATION NATION
ON JOIN_DATA.S_NATIONKEY = NATION.N_NATIONKEY
LEFT JOIN TESE_DATABASE.SA.REGION REGION
ON NATION.N_REGIONKEY = REGION.R_REGIONKEY);
`;
var statement = snowflake.createStatement({
sqlText: query_text
});
statement.execute();
';

CREATE OR REPLACE PROCEDURE TESE_DATABASE.SADL."SADL_FACT_LINEORDER_LOAD" ()
RETURNS VARCHAR(16777216)
LANGUAGE JAVASCRIPT
EXECUTE AS OWNER
AS '
var query_text = `
INSERT INTO TESE_DATABASE.SADL.FACT_LINEORDER (
SELECT
LINEITEM.L_ORDERKEY AS LO_ORDERKEY,
LINEITEM.L_LINENUMBER AS LO_LINENUMBER,
ORDERS.O_CUSTKEY AS LO_CUSTKEY,
LINEITEM.L_PARTKEY AS LO_PARTKEY,
LINEITEM.L_SUPPKEY AS LO_SUPPKEY,
ORDERS.O_ORDERDATE AS LO_ORDERDATE,
ORDERS.O_ORDERPRIORITY AS LO_ORDERPRIORITY,
ORDERS.O_SHIPPRIORITY AS LO_SHIPPRIORITY,
LINEITEM.L_QUANTITY AS LO_QUANTITY,
LINEITEM.L_EXTENDEDPRICE AS LO_EXTENDEDPRICE,
ORDERS.O_TOTALPRICE AS LO_TOTALPRICE,
LINEITEM.L_DISCOUNT AS LO_DISCOUNT,
LINEITEM.L_EXTENDEDPRICE * (100 - LINEITEM.L_DISCOUNT) / 100 AS
LO_REVENUE,
PARTSUPP.PS_SUPPLYCOST AS LO_SUPPLYCOST,
LINEITEM.L_TAX AS LO_TAX,
LINEITEM.L_COMMITDATE AS LO_COMMITDATE,
LINEITEM.L_SHIPMODE
FROM TESE_DATABASE.SA.LINEITEM LINEITEM
LEFT JOIN TESE_DATABASE.SA.ORDERS ORDERS
ON LINEITEM.L_ORDERKEY = ORDERS.O_ORDERKEY
LEFT JOIN TESE_DATABASE.SA.PARTSUPP PARTSUPP
ON PARTSUPP.PS_PARTKEY = LINEITEM.L_PARTKEY);
`;
var statement = snowflake.createStatement({

```

```

    sqlText: query_text
  });
  statement.execute();
';

```

De seguida, encontra-se o código utilizado na plataforma Google BigQuery.

```

CREATE OR REPLACE PROCEDURE SADL.SADL_DIM_CUSTOMER_LOAD()
BEGIN
INSERT INTO engaged-mariner-345217.SADL.DIM_CUSTOMER (
WITH JOIN_DATA AS (
  SELECT *
  FROM engaged-mariner-345217.SA.customer
  CROSS JOIN UNNEST(['1', '2', '3', '4', '5', '6', '7', '8', '9']) AS
counter_9
  ORDER BY C_CUSTKEY
)
  SELECT
    JOIN_DATA.C_CUSTKEY,
    JOIN_DATA.C_NAME,
    JOIN_DATA.C_ADDRESS,
    CONCAT(LEFT(NATION.N_NAME, 9), JOIN_DATA.counter_9) AS C_CITY,
    NATION.N_NAME AS C_NATION,
    REGION.R_NAME AS C_REGION,
    JOIN_DATA.C_PHONE,
    JOIN_DATA.C_MKTSEGMENT
FROM JOIN_DATA
LEFT JOIN engaged-mariner-345217.SA.nation NATION
  ON JOIN_DATA.C_NATIONKEY = NATION.N_NATIONKEY
LEFT JOIN engaged-mariner-345217.SA.region REGION
  ON NATION.N_REGIONKEY = REGION.R_REGIONKEY);
END
;

```

```

CREATE OR REPLACE PROCEDURE SADL.SADL_DIM_DATE_LOAD()
BEGIN
INSERT INTO engaged-mariner-345217.SADL.DIM_DATE (
  SELECT
    date AS d_datekey,
    CAST(date AS STRING) AS d_date,
    CASE
      WHEN EXTRACT(DAYOFWEEK FROM date)=1 THEN 'Sunday'
      WHEN EXTRACT(DAYOFWEEK FROM date)=2 THEN 'Monday'
      WHEN EXTRACT(DAYOFWEEK FROM date)=3 THEN 'Tuesday'
      WHEN EXTRACT(DAYOFWEEK FROM date)=4 THEN 'Wednesday'
      WHEN EXTRACT(DAYOFWEEK FROM date)=5 THEN 'Thursday'
      WHEN EXTRACT(DAYOFWEEK FROM date)=6 THEN 'Friday'
      WHEN EXTRACT(DAYOFWEEK FROM date)=7 THEN 'Saturday'
    END AS d_dayofweek,
    CASE
      WHEN EXTRACT(MONTH FROM date)=1 THEN 'January'
      WHEN EXTRACT(MONTH FROM date)=2 THEN 'February'
      WHEN EXTRACT(MONTH FROM date)=3 THEN 'March'
      WHEN EXTRACT(MONTH FROM date)=4 THEN 'April'
      WHEN EXTRACT(MONTH FROM date)=5 THEN 'May'

```

```

        WHEN EXTRACT (MONTH FROM date)=6 THEN 'June'
        WHEN EXTRACT (MONTH FROM date)=7 THEN 'July'
        WHEN EXTRACT (MONTH FROM date)=8 THEN 'August'
        WHEN EXTRACT (MONTH FROM date)=9 THEN 'September'
        WHEN EXTRACT (MONTH FROM date)=10 THEN 'October'
        WHEN EXTRACT (MONTH FROM date)=11 THEN 'November'
        WHEN EXTRACT (MONTH FROM date)=12 THEN 'December'
    END AS d_month,
    EXTRACT (YEAR FROM date) AS d_year,
    EXTRACT (YEAR FROM date) || LPAD (CAST (EXTRACT (MONTH FROM date) AS STRING),
2, '0') AS d_yearmonthnum,
    SUBSTRING (FORMAT_DATETIME ("%B", DATETIME (date)), 0, 3) || EXTRACT (YEAR
FROM date) AS d_yearmonth,
    EXTRACT (DAYOFWEEK FROM date) AS d_daynuminweek,
    CASE
        WHEN EXTRACT (YEAR FROM date) = 1992 THEN EXTRACT (WEEK FROM date) + 1
        WHEN EXTRACT (YEAR FROM date) = 1996 THEN EXTRACT (WEEK FROM date) + 1
        WHEN EXTRACT (YEAR FROM date) = 1997 THEN EXTRACT (WEEK FROM date) + 1
        ELSE EXTRACT (WEEK FROM date)
    END AS d_weeknuminyear,
    FROM UNNEST (GENERATE_DATE_ARRAY ('1992-01-01', '1997-12-31')) AS date
    ORDER BY d_datekey);
END
;

CREATE OR REPLACE PROCEDURE SADL.SADL_DIM_PART_LOAD ()
BEGIN
    INSERT INTO engaged-mariner-345217.SADL.DIM_PART (
        WITH JOIN_DATA AS (
            SELECT *
            FROM engaged-mariner-345217.SA.parts
            CROSS JOIN UNNEST ([ '1', '2', '3', '4', '5', '6', '7', '8', '9', '10',
'11', '12', '13', '14', '15', '16', '17', '18',
'19', '20',
'21', '22', '23', '24', '25', '26', '27', '28',
'29', '30',
'31', '32', '33', '34', '35', '36', '37', '38',
'39', '40']) AS counter_9
            ORDER BY P_PARTKEY
        )
        SELECT
            JOIN_DATA.P_PARTKEY,
            SUBSTRING (P_NAME, 0, INSTR (P_NAME, ' ') + 1 + INSTR (SUBSTR (P_NAME,
INSTR (P_NAME, ' ') + 2), ' ')) AS P_NAME,
            REPLACE (JOIN_DATA.P_MFGR, 'Manufacturer', 'MFGR') AS P_MFGR,
            REPLACE (JOIN_DATA.P_BRAND, 'Brand', 'MFGR') AS P_CATEGORY,
            REPLACE (CONCAT (JOIN_DATA.P_BRAND, JOIN_DATA.counter_9), 'Brand',
'MFGR') AS P_BRAND1,
            JOIN_DATA.P_NAME AS P_COLOR,
            JOIN_DATA.P_TYPE,
            JOIN_DATA.P_SIZE,
            JOIN_DATA.P_CONTAINER
        FROM JOIN_DATA
        ORDER BY P_PARTKEY, P_BRAND1);
END
;

```

```

CREATE OR REPLACE PROCEDURE SADL.SADL_DIM_SUPPLIER_LOAD()
BEGIN
INSERT INTO engaged-mariner-345217.SADL.DIM_SUPPLIER(
WITH JOIN_DATA AS (
    SELECT *
    FROM engaged-mariner-345217.SA.supplier
    CROSS JOIN UNNEST(['1', '2', '3', '4', '5', '6', '7', '8', '9']) AS
counter_9
    ORDER BY S_SUPPKEY
)
SELECT
    JOIN_DATA.S_SUPPKEY,
    JOIN_DATA.S_NAME,
    JOIN_DATA.S_ADDRESS,
    CONCAT(LEFT(NATION.N_NAME, 9), JOIN_DATA.counter_9) AS S_CITY,
    NATION.N_NAME AS S_NATION,
    REGION.R_NAME AS S_REGION,
    JOIN_DATA.S_PHONE
FROM JOIN_DATA
LEFT JOIN engaged-mariner-345217.SA.nation NATION
    ON JOIN_DATA.S_NATIONKEY = NATION.N_NATIONKEY
LEFT JOIN engaged-mariner-345217.SA.region REGION
    ON NATION.N_REGIONKEY = REGION.R_REGIONKEY);
END
;

CREATE OR REPLACE PROCEDURE SADL.SADL_FACT_LINEORDER_LOAD()
BEGIN
INSERT INTO engaged-mariner-345217.SADL.FACT_LINEORDER (
SELECT
    LINEITEM.L_ORDERKEY AS LO_ORDERKEY,
    LINEITEM.L_LINENUMBER AS LO_LINENUMBER,
    ORDERS.O_CUSTKEY AS LO_CUSTKEY,
    LINEITEM.L_PARTKEY AS LO_PARTKEY,
    LINEITEM.L_SUPPKEY AS LO_SUPPKEY,
    ORDERS.O_ORDERDATE AS LO_ORDERDATE,
    ORDERS.O_ORDERPRIORITY AS LO_ORDERPRIORITY,
    ORDERS.O_SHIPPRIORITY AS LO_SHIPPRIORITY,
    LINEITEM.L_QUANTITY AS LO_QUANTITY,
    LINEITEM.L_EXTENDEDPRICE AS LO_EXTENDEDPRICE,
    ORDERS.O_TOTALPRICE AS LO_TOTALPRICE,
    LINEITEM.L_DISCOUNT AS LO_DISCOUNT,
    ROUND(LINEITEM.L_EXTENDEDPRICE * (100 - LINEITEM.L_DISCOUNT) / 100, 9)
AS LO_REVENUE,
    PARTSUPP.PS_SUPPLYCOST AS LO_SUPPLYCOST,
    LINEITEM.L_TAX AS LO_TAX,
    LINEITEM.L_COMMITDATE AS LO_COMMITDATE,
    LINEITEM.L_SHIPMODE
FROM engaged-mariner-345217.SA.lineitem LINEITEM
LEFT JOIN engaged-mariner-345217.SA.orders ORDERS
    ON LINEITEM.L_ORDERKEY = ORDERS.O_ORDERKEY
LEFT JOIN engaged-mariner-345217.SA.partsupp PARTSUPP
    ON PARTSUPP.PS_PARTKEY = LINEITEM.L_PARTKEY);
END
;

```


A seguir, encontra-se o código utilizado na plataforma Amazon Redshift.

```
CREATE OR REPLACE PROCEDURE TESE_DATABASE.SADL."SADL_DIM_CUSTOMER_LOAD" ()
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO TESE_DATABASE.SADL.DIM_CUSTOMER (
        WITH recursive numbers(NUMBER) AS (
            SELECT 1
            UNION ALL
            SELECT NUMBER + 1
            FROM numbers
            WHERE NUMBER < 9
        ),
        JOIN_DATA AS (
            SELECT *
            FROM TESE_DATABASE.SA.CUSTOMER
            CROSS JOIN numbers
        )
    SELECT
        JOIN_DATA.C_CUSTKEY,
        JOIN_DATA.C_NAME,
        JOIN_DATA.C_ADDRESS,
        CONCAT(LEFT(NATION.N_NAME, 9), JOIN_DATA.number) AS C_CITY,
        NATION.N_NAME AS C_NATION,
        REGION.R_NAME AS C_REGION,
        JOIN_DATA.C_PHONE,
        JOIN_DATA.C_MKTSEGMENT
    FROM JOIN_DATA
    LEFT JOIN TESE_DATABASE.SA.NATION NATION
        ON JOIN_DATA.C_NATIONKEY = NATION.N_NATIONKEY
    LEFT JOIN TESE_DATABASE.SA.REGION REGION
        ON NATION.N_REGIONKEY = REGION.R_REGIONKEY);
END;
$$
```

```
CREATE OR REPLACE PROCEDURE TESE_DATABASE.SADL."SADL_DIM_DATE_LOAD" ()
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO TESE_DATABASE.SADL.DIM_DATE (
        WITH recursive t(n) AS (
            SELECT 1::integer
            UNION all
            SELECT n+1
            FROM t
            WHERE n < 2192
        ),
        CTE_MY_DATE AS (
            SELECT '1991-12-31' + n AS MY_DATE
            FROM t
        )
    SELECT
        TO_CHAR(MY_DATE, 'YYYY-MM-DD') AS d_datekey,
```

```

TO_CHAR(MY_DATE, 'YYYY-MM-DD') AS d_date,
TO_CHAR(MY_DATE, 'Day') AS d_dayofweek,
TO_CHAR(MY_DATE, 'Month') AS d_month,
EXTRACT(YEAR FROM MY_DATE) AS d_year,
CASE
    WHEN EXTRACT(MONTH FROM MY_DATE) BETWEEN 1 AND 9
    THEN CAST(EXTRACT(YEAR FROM MY_DATE) AS VARCHAR) || '0' ||
CAST(EXTRACT(MONTH FROM MY_DATE) AS VARCHAR)
    ELSE EXTRACT(YEAR FROM MY_DATE) || EXTRACT(MONTH FROM MY_DATE)
END AS d_yearmonthnum,
INITCAP(TO_CHAR(MY_DATE, 'MONYYYY')) AS d_yearmonth,
DATE_PART(dow, MY_DATE) + 1 AS d_daynuminweek,
DATE_PART(d, MY_DATE) AS d_daynuminmonth,
DATE_PART(doy, MY_DATE) AS d_daynuminyear,
CAST(EXTRACT(MONTH FROM MY_DATE) AS INT) AS d_monthnuminyear,
CASE
    WHEN DATE_PART(dow, MY_DATE) = '0' THEN DATE_PART(w, MY_DATE) + 1
    ELSE DATE_PART(w, MY_DATE)
END AS d_weeknuminyear,
CASE
    WHEN d_daynuminweek = '7' THEN 'Yes'
    ELSE 'No'
END AS d_lastdayinweekfl,
CASE
    WHEN d_daynuminmonth = '31' THEN 'Yes'
    WHEN (d_monthnuminyear = 4 OR d_monthnuminyear = 6 OR
d_monthnuminyear = 9 OR d_monthnuminyear = 11)
    AND d_daynuminmonth = '30' THEN 'Yes'
    WHEN (d_year = '1992' OR d_year = '1996') AND d_monthnuminyear =
2 AND d_daynuminmonth = '29' THEN 'Yes'
    WHEN (d_year = '1993' OR d_year = '1994' OR d_year = '1995' OR
d_year = '1997')
    AND (d_monthnuminyear = 2) AND d_daynuminmonth = '28' THEN
'Yes'
    ELSE 'No'
END AS d_lastdayinmonthfl,
CASE
    WHEN d_dayofweek = 'Saturday' OR d_dayofweek = 'Sunday' THEN
'Weekend'
    ELSE 'Weekday'
END AS d_weekdayfl
FROM CTE_MY_DATE
ORDER BY d_datekey
);
END;
$$

```

```

CREATE OR REPLACE PROCEDURE TESE_DATABASE.SADL."SADL_DIM_PART_LOAD" ()
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO TESE_DATABASE.SADL.DIM_PART (
        WITH recursive numbers(NUMBER) AS (
            SELECT 1
            UNION ALL
            SELECT NUMBER + 1

```

```

        FROM numbers
        WHERE NUMBER < 40
    ),
    JOIN_DATA AS (
        SELECT *
        FROM TESE_DATABASE.SA.PART
        CROSS JOIN numbers
        ORDER BY P_PARTKEY
    )
    SELECT
        JOIN_DATA.P_PARTKEY,
        SUBSTRING(P_NAME, 0, REGEXP_INSTR(P_NAME, ' ', 2, 2)) AS P_NAME,
        REPLACE(JOIN_DATA.P_MFGR, 'Manufacturer', 'MFGR') AS P_MFGR,
        REPLACE(JOIN_DATA.P_BRAND, 'Brand', 'MFGR') AS P_CATEGORY,
        REPLACE(CONCAT(JOIN_DATA.P_BRAND, JOIN_DATA.number), 'Brand',
'MFGR') AS P_BRAND1,
        JOIN_DATA.P_NAME AS P_COLOR,
        JOIN_DATA.P_TYPE,
        JOIN_DATA.P_SIZE,
        JOIN_DATA.P_CONTAINER
    FROM JOIN_DATA
        ORDER BY P_PARTKEY, P_BRAND1);
END;
$$

CREATE OR REPLACE PROCEDURE TESE_DATABASE.SADL."SADL_DIM_SUPPLIER_LOAD" ()
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO TESE_DATABASE.SADL.DIM_SUPPLIER (

        WITH recursive numbers (NUMBER) AS (
            SELECT 1
            UNION ALL
            SELECT NUMBER + 1
            FROM numbers
            WHERE NUMBER < 9
        ),
        JOIN_DATA AS (
            SELECT *
            FROM TESE_DATABASE.SA.SUPPLIER
            CROSS JOIN numbers
        )
        SELECT
            JOIN_DATA.S_SUPPKEY,
            JOIN_DATA.S_NAME,
            JOIN_DATA.S_ADDRESS,
            CONCAT(LEFT(NATION.N_NAME, 9), JOIN_DATA.number) AS S_CITY,
            NATION.N_NAME AS S_NATION,
            REGION.R_NAME AS S_REGION,
            JOIN_DATA.S_PHONE
        FROM JOIN_DATA
        LEFT JOIN TESE_DATABASE.SA.NATION NATION
            ON JOIN_DATA.S_NATIONKEY = NATION.N_NATIONKEY
        LEFT JOIN TESE_DATABASE.SA.REGION REGION
            ON NATION.N_REGIONKEY = REGION.R_REGIONKEY);

```

```

END;
$$

CREATE OR REPLACE PROCEDURE TESE_DATABASE.SADL."SADL_FACT_LINEORDER_LOAD" ()
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO TESE_DATABASE.SADL.FACT_LINEORDER (
        SELECT
            LINEITEM.L_ORDERKEY AS LO_ORDERKEY,
            LINEITEM.L_LINENUMBER AS LO_LINENUMBER,
            ORDERS.O_CUSTKEY AS LO_CUSTKEY,
            LINEITEM.L_PARTKEY AS LO_PARTKEY,
            LINEITEM.L_SUPPKEY AS LO_SUPPKEY,
            ORDERS.O_ORDERDATE AS LO_ORDERDATE,
            ORDERS.O_ORDERPRIORITY AS LO_ORDERPRIORITY,
            ORDERS.O_SHIPPRIORITY AS LO_SHIPPRIORITY,
            LINEITEM.L_QUANTITY AS LO_QUANTITY,
            LINEITEM.L_EXTENDEDPRICE AS LO_EXTENDEDPRICE,
            ORDERS.O_TOTALPRICE AS LO_TOTALPRICE,
            LINEITEM.L_DISCOUNT AS LO_DISCOUNT,
            ROUND(LINEITEM.L_EXTENDEDPRICE * (100 - LINEITEM.L_DISCOUNT) / 100,
15) AS LO_REVENUE,
            PARTSUPP.PS_SUPPLYCOST AS LO_SUPPLYCOST,
            LINEITEM.L_TAX AS LO_TAX,
            LINEITEM.L_COMMITDATE AS LO_COMMITDATE,
            LINEITEM.L_SHIPMODE
        FROM TESE_DATABASE.SA.LINEITEM LINEITEM
        LEFT JOIN TESE_DATABASE.SA.ORDERS ORDERS
            ON LINEITEM.L_ORDERKEY = ORDERS.O_ORDERKEY
        LEFT JOIN TESE_DATABASE.SA.PARTSUPP PARTSUPP
            ON PARTSUPP.PS_PARTKEY = LINEITEM.L_PARTKEY);
    END;
$$

```

Por fim, é a seguir apresentado o código utilizado na plataforma Azure Synapse.

```

CREATE PROCEDURE SADL.SADL_DIM_CUSTOMER_LOAD
AS
BEGIN
    WITH NATION_NAME_MODIF AS (
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '1') AS Name FROM SA.NATION
    UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '2') AS Name FROM SA.NATION
    UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '3') AS Name FROM SA.NATION
    UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '4') AS Name FROM SA.NATION
    UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '5') AS Name FROM SA.NATION
    UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '6') AS Name FROM SA.NATION
    UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '7') AS Name FROM SA.NATION
    UNION ALL

```

```

        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '8') AS Name FROM SA.NATION
UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '9') AS Name FROM SA.NATION
    )
    INSERT INTO SADL.DIM_CUSTOMER
    SELECT
        CUSTOMER.C_CUSTKEY,
        CUSTOMER.C_NAME,
        CUSTOMER.C_ADDRESS,
        NATION_NAME_MODIF.Name AS C_CITY,
        NATION.N_NAME AS C_NATION,
        REGION.R_NAME AS C_REGION,
        CUSTOMER.C_PHONE,
        CUSTOMER.C_MKTSEGMENT
    FROM SA.customer CUSTOMER
    LEFT JOIN SA.NATION NATION
        ON CUSTOMER.C_NATIONKEY = NATION.N_NATIONKEY
    LEFT JOIN SA.REGION REGION
        ON NATION.N_REGIONKEY = REGION.R_REGIONKEY
    LEFT JOIN NATION_NAME_MODIF
        ON NATION.N_NATIONKEY = NATION_NAME_MODIF.N_NATIONKEY;
END;

CREATE PROCEDURE SADL.SADL_DIM_DATE_LOAD
AS
BEGIN
    DECLARE @StartDate DATE = '1992-01-01', @NumberOfYears INT = 6;
    DECLARE @CutoffDate DATE = DATEADD(YEAR, @NumberOfYears, @StartDate);
    WITH DATE_TEMPORARY_TABLE AS (
        SELECT LEFT(d, 10) AS dates
        FROM (
            SELECT d = DATEADD(DAY, rn - 1, @StartDate)
            FROM (
                SELECT TOP (DATEDIFF(DAY, @StartDate, @CutoffDate))
                rn = ROW_NUMBER() OVER (ORDER BY s1.[object_id])
                FROM sys.all_objects AS s1
                CROSS JOIN sys.all_objects AS s2
                ORDER BY s1.[object_id]
            ) AS x
        ) AS y
    )
    INSERT INTO SADL.DIM_DATE
    SELECT
        dates AS d_datekey,
        dates AS d_date,
        DATENAME(WEEKDAY, dates) AS d_dayofweek,
        DATENAME(MONTH, dates) AS d_month,
        DATEPART(YEAR, dates) AS d_year,
        CASE
            WHEN DATEPART(MONTH, dates) BETWEEN 1 AND 9
            THEN CONCAT(DATEPART(YEAR, dates), '0', DATEPART(MONTH,
dates))
            ELSE CONCAT(DATEPART(YEAR, dates), DATEPART(MONTH, dates))
        END AS d_yearmonthnum,
        CONCAT(LEFT(DATENAME(MONTH, dates), 3), DATEPART(YEAR, dates)) AS
d_yearmonth,

```

```

DATEPART(WEEKDAY, dates) AS d_daynuminweek,
DATEPART(DAY, dates) AS d_daynuminmonth,
DATEPART(DAYOFYEAR, dates) AS d_daynuminyear,
DATEPART(MONTH, dates) AS d_monthnuminyear,
CASE
    WHEN DATEPART(YEAR, dates) = '1993' THEN DATEPART(WEEK, dates)
- 1
    WHEN DATEPART(YEAR, dates) = '1994' THEN DATEPART(WEEK, dates)
- 1
    ELSE DATEPART(WEEK, dates)
END AS d_weeknuminyear,
CASE
    WHEN DATEPART(WEEKDAY, dates) = '7' THEN 'Yes'
    ELSE 'No'
END AS d_lastdayinweekfl,
CASE
    WHEN DATEPART(DAY, dates) = '31' THEN 'Yes'
    WHEN (DATEPART(MONTH, dates) = 4 OR DATEPART(MONTH, dates) = 6
OR DATEPART(MONTH, dates) = 9 OR DATEPART(MONTH, dates) = 11)
    AND DATEPART(DAY, dates) = '30' THEN 'Yes'
    WHEN (DATEPART(YEAR, dates) = '1992' OR DATEPART(YEAR, dates) =
'1996') AND DATEPART(MONTH, dates) = 2 AND DATEPART(DAY, dates) = '29' THEN
'Yes'
    WHEN (DATEPART(YEAR, dates) = '1993' OR DATEPART(YEAR, dates) =
'1994' OR DATEPART(YEAR, dates) = '1995' OR DATEPART(YEAR, dates) = '1997')
    AND (DATEPART(MONTH, dates) = 2) AND DATEPART(DAY, dates) =
'28' THEN 'Yes'
    ELSE 'No'
END AS d_lastdayinmonthfl,
CASE
    WHEN DATENAME(WEEKDAY, dates) = 'Saturday' OR DATENAME(WEEKDAY,
dates) = 'Sunday' THEN 'Weekend'
    ELSE 'Weekday'
END AS d_weekdayfl
FROM DATE_TEMPORARY_TABLE;
END;

```

```

CREATE PROCEDURE SADL.SADL_DIM_PART_LOAD
AS
BEGIN
    WITH NATION_NAME_MODIF AS (
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '1') AS Name
        FROM SA.NATION
        UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '2') AS Name
        FROM SA.NATION
        UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '3') AS Name
        FROM SA.NATION
        UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '4') AS Name
        FROM SA.NATION
        UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '5') AS Name
        FROM SA.NATION
        UNION ALL

```

```

SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '6') AS Name
FROM SA.NATION
UNION ALL
SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '7') AS Name
FROM SA.NATION
UNION ALL
SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '8') AS Name
FROM SA.NATION
UNION ALL
SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '9') AS Name
FROM SA.NATION
)
INSERT INTO SADL.DIM_CUSTOMER
SELECT
    CUSTOMER.C_CUSTKEY,
    CUSTOMER.C_NAME,
    CUSTOMER.C_ADDRESS,
    NATION_NAME_MODIF.Name AS C_CITY,
    NATION.N_NAME AS C_NATION,
    REGION.R_NAME AS C_REGION,
    CUSTOMER.C_PHONE,
    CUSTOMER.C_MKTSEGMENT
FROM SA.customer CUSTOMER
LEFT JOIN SA.NATION NATION
    ON CUSTOMER.C_NATIONKEY = NATION.N_NATIONKEY
LEFT JOIN SA.REGION REGION
    ON NATION.N_REGIONKEY = REGION.R_REGIONKEY
LEFT JOIN NATION_NAME_MODIF
    ON NATION.N_NATIONKEY = NATION_NAME_MODIF.N_NATIONKEY;
END;

CREATE PROCEDURE SADL.SADL_DIM_SUPPLIER_LOAD
AS
BEGIN
    WITH NATION_NAME_MODIF AS (
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '1') AS Name FROM SA.NATION
    UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '2') AS Name FROM SA.NATION
    UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '3') AS Name FROM SA.NATION
    UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '4') AS Name FROM SA.NATION
    UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '5') AS Name FROM SA.NATION
    UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '6') AS Name FROM SA.NATION
    UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '7') AS Name FROM SA.NATION
    UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '8') AS Name FROM SA.NATION
    UNION ALL
        SELECT N_NATIONKEY, (LEFT(N_NAME, 9) + '9') AS Name FROM SA.NATION
    )
    INSERT INTO SADL.DIM_SUPPLIER
    SELECT
        SUPPLIER.S_SUPPKEY,

```

```

        SUPPLIER.S_NAME,
        SUPPLIER.S_ADDRESS,
        NATION_NAME_MODIF.Name AS S_CITY,
        NATION.N_NAME AS S_NATION,
        REGION.R_NAME AS S_REGION,
        SUPPLIER.S_PHONE
    FROM SA.supplier SUPPLIER
    LEFT JOIN SA.NATION NATION
        ON SUPPLIER.S_NATIONKEY = NATION.N_NATIONKEY
    LEFT JOIN SA.REGION REGION
        ON NATION.N_REGIONKEY = REGION.R_REGIONKEY
    LEFT JOIN NATION_NAME_MODIF
        ON NATION.N_NATIONKEY = NATION_NAME_MODIF.N_NATIONKEY;
END;

CREATE PROCEDURE SADL.SADL_FACT_LINEORDER_LOAD
AS
BEGIN
    INSERT INTO SADL.FACT_LINEORDER
    SELECT
        LINEITEM.L_ORDERKEY AS LO_ORDERKEY,
        LINEITEM.L_LINENUMBER AS LO_LINENUMBER,
        ORDERS.O_CUSTKEY AS LO_CUSTKEY,
        LINEITEM.L_PARTKEY AS LO_PARTKEY,
        LINEITEM.L_SUPPKEY AS LO_SUPPKEY,
        ORDERS.O_ORDERDATE AS LO_ORDERDATE,
        ORDERS.O_ORDERPRIORITY AS LO_ORDERPRIORITY,
        ORDERS.O_SHIPPRIORITY AS LO_SHIPPRIORITY,
        LINEITEM.L_QUANTITY AS LO_QUANTITY,
        LINEITEM.L_EXTENDEDPRICE AS LO_EXTENDEDPRICE,
        ORDERS.O_TOTALPRICE AS LO_TOTALPRICE,
        LINEITEM.L_DISCOUNT AS LO_DISCOUNT,
        ROUND(LINEITEM.L_EXTENDEDPRICE * (100 - LINEITEM.L_DISCOUNT) / 100, 15)
    AS LO_REVENUE,
        PARTSUPP.PS_SUPPLYCOST AS LO_SUPPLYCOST,
        LINEITEM.L_TAX AS LO_TAX,
        LINEITEM.L_COMMITDATE AS LO_COMMITDATE,
        LINEITEM.L_SHIPMODE
    FROM SA.LINEITEM LINEITEM
    LEFT JOIN SA.ORDERS ORDERS
        ON LINEITEM.L_ORDERKEY = ORDERS.O_ORDERKEY
    LEFT JOIN SA.PARTSUPP PARTSUPP
        ON PARTSUPP.PS_PARTKEY = LINEITEM.L_PARTKEY;
END;

```


ANEXOS

Anexo A – Estrutura das Tabelas do Schema TPC-H

Tabela PART

Nome da Coluna	Tipo de Dados	Comentários
P_PARTKEY	Identifier	Primary Key
P_NAME	Variable text, size 55	
P_MFGR	Fixed text, size 25	
P_BRAND	Fixed text, size 10	
P_TYPE	Variable text, size 25	
P_SIZE	Integer	
P_CONTAINER	Fixed text, size 10	
P_RETAILPRICE	Decimal	
P_COMMENT	Variable text, size 23	

Tabela 16 - Estrutura da Tabela PART

Tabela SUPPLIER

Nome da Coluna	Tipo de Dados	Comentários
S_SUPPKEY	Identifier	Primary Key
S_NAME	Fixed text, size 25	
S_ADDRESS	Variable text, size 40	
S_NATIONKEY	Identifier	Foreign Key para a N_NATIONKEY na tabela NATION
S_PHONE	Fixed text, size 15	
S_ACCTBAL	Decimal	
S_COMMENT	Variable text, size 101	

Tabela 17 - Estrutura da Tabela SUPPLIER

Tabela PARTSUPP

Nome da Coluna	Tipo de Dados	Comentários
PS_PARTKEY	Identifier	Primary Key / Foreign Key para P_PARTKEY na tabela PART
PS_SUPPKEY	Identifier	Primary Key / Foreign Key para S_SUPPKEY na tabela SUPPLIER
PS_AVAILQTY	Integer	
PS_SUPPLYCOST	Decimal	
PS_COMMENT	Variable text, size 199	

*Tabela 18 - Estrutura da Tabela PARTSUPP***Tabela CUSTOMER**

Nome da Coluna	Tipo de Dados	Comentários
C_CUSTKEY	Identifier	Primary Key
C_NAME	Variable text, size 25	
C_ADDRESS	Variable text, size 40	
C_NATIONKEY	Identifier	Foreign Key para N_NATIONKEY na tabela NATION
C_PHONE	Fixed text, size 15	
C_ACCTBAL	Decimal	
C_MKTSEGMENT	Fixed text, size 10	
C_COMMENT	Variable text, size 117	

*Tabela 19 - Estrutura da Tabela CUSTOMER***Tabela NATION**

Nome da Coluna	Tipo de Dados	Comentários
N_NATIONKEY	Identifier	Primary Key
N_NAME	Fixed text, size 25	
N_REGIONKEY	Identifier	Foreign Key para R_REGIONKEY na tabela SUPPLIERREGIONR
N_COMMENT	Fixed text, size 152	

Tabela 20 - Estrutura da Tabela NATION

Tabela ORDERS

Nome da Coluna	Tipo de Dados	Comentários
O_ORDERKEY	Identifier	Primary Key
O_CUSTKEY	Identifier	Foreign Key para C_CUSTKEY na tabela CUSTOMER
O_ORDERSTATUS	Fixed text, size 1	
O_TOTALPRICE	Decimal	
O_ORDERDATE	Date	
O_ORDERPRIORITY	Fixed text, size 15	
O_CLERK	Fixed text, size 15	
O_SHIPPRIORITY	Integer	
O_COMMENT	VARIABLE text, size 79	

Tabela 21 - Estrutura da Tabela ORDERS

Tabela LINEITEM

Nome da Coluna	Tipo de Dados	Comentários
L_ORDERKEY	Identifier	Primary Key / Foreign Key para O_ORDERKEY na tabela ORDERS
L_PARTKEY	Identifier	Foreign Key para P_PARTKEY na tabela PART
L_SUPPKEY	Identifier	Foreign Key para S_SUPPKEY na tabela SUPPLIER
L_LINENUMBER	Integer	Primary Key
L_QUANTITY	Decimal	
L_EXTENDEDPRICE	Decimal	
L_DISCOUNT	Decimal	
L_TAX	Decimal	
L_RETURNFLAG	Fixed text, size 1	
L_LINESTATUS	Fixed text, size 1	
L_SHIPDATE	Date	
L_COMMITDATE	Date	
L_RECEIPTDATE	Date	
L_SHIPINSTRUCT	Fixed text, size 25	
L_SHIPMODE	Fixed text, size 10	
L_COMMENT	Variable text, size 44	

Tabela 22 - Estrutura da Tabela LINEITEM

Tabela REGION

Nome da Coluna	Tipo de Dados	Comentários
R_REGIONKEY	Identifier	Primary Key
R_NAME	Fixed text, size 25	
R_COMMENT	Variable text, size 152	

Tabela 23 - Estrutura da Tabela REGION

Anexo B – Construção da Base de Dados e da Staging Area

De forma a inicializar o uso do “Espaço de trabalho SQL” no BigQuery, é necessário primeiramente criar um novo projeto, no qual foi criado um denominado “My First Project” e ao qual o BigQuery atribuiu o id “engaged-mariner-345217”. De seguida, basta seleccionar o projeto e “Criar um Conjunto de Dados”, de modo a criar o schema denominado SA. Estes passos são demonstrados nas figuras 13 e 14.

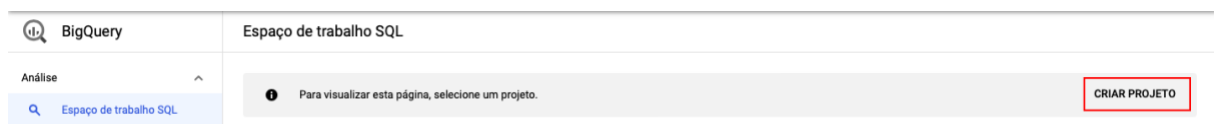


Figura 13 - Criação de um novo Projeto no BigQuery



Explorador

Editor - x +

EXECUTAR

1

Criar conjunto de dados

Criar conjunto de dados

ID do projeto
engaged-mariner-345217 MUDAR

Código do conjunto de dados *
SA
Letras, números e sublinhados são permitidos

Local dos dados

Expiração da tabela padrão
 Ativar expiração da tabela
Idade máxima padrão da tabela Days

Opções avançadas

CRIAR CONJUNTO DE DADOS CANCELAR

Figura 14 - Criação da SA no BigQuery

Ao aceder ao Amazon Redshift na AWS, é necessário abrir o menu do lado esquerdo e seleccionar “Query Editor” ou “Query Editor v.2” de modo a aceder à página onde o trabalho

será realizado. Na área “Databases” após selecionar o cluster em que se quer trabalhar, basta criar uma base de dados e, posteriormente, um schema, tal como no Snowflake. Estes passos são demonstrados nas figuras 15 e 16.

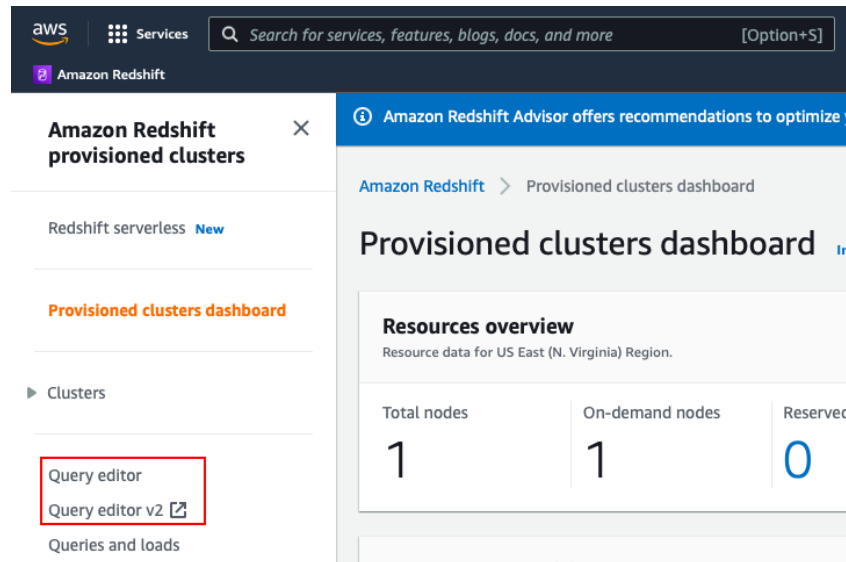


Figura 15 - Acesso ao Query Editor no Redshift

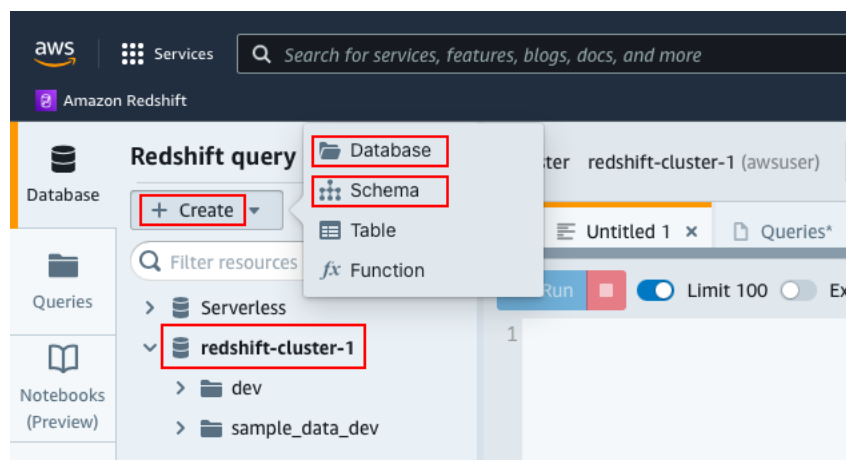


Figura 16 - Criação da Base de Dados e da SA no Redshift

De modo a inicializar a utilização do Azure Synapse, é necessário primeira criar uma “Área de Trabalho da Synapse”, acedendo à página apresentada na figura 17, e preenchendo os campos necessários, sendo aqui também criado um Data Lake Storage (DLS). Após a sua criação, deve-se aceder à mesma e clicar em “Abrir Synapse Studio”, seguido de aceder à sua área “Gerir”, e criar um conjunto de SQL dedicado, sendo este ambiente onde se criará os *schemas* e as tabelas do modelo. Estes passos estão ilustrados na figura 18. Posteriormente,

de modo a criar o *schema* da SA, é apenas necessário correr o comando “CREATE SCHEMA SA” neste novo ambiente.

The image consists of two screenshots from the Microsoft Azure portal. The top screenshot shows the 'Azure Synapse Analytics' page for 'Universidade do Minho'. A red box highlights the '+ Criar' button. Below it, a table lists existing workspaces, with one entry 'tесеareadetrabalho' highlighted in a red box. The bottom screenshot shows the 'Criar área de trabalho do Synapse' form. It has tabs for 'Informações básicas', 'Segurança', 'Rede', 'Etiquetas', and 'Rever + criar'. The 'Informações básicas' tab is active. It contains sections for 'Detalhes do projeto' and 'Detalhes da área de trabalho'. In the 'Detalhes do projeto' section, 'Subscrição' is set to 'Azure for Students', 'Grupo de recursos' is 'tese_grupo_recursos', and 'Grupo de recursos geridos' is 'Introduza o nome do grupo de recursos geridos'. In the 'Detalhes da área de trabalho' section, 'Nome da área de trabalho' is 'teseareadetrabalho', 'Região' is 'West Europe', 'Seleção Data Lake Storage Gen2' is 'Da subscrição', 'Nome da conta' is 'tesedatalakestorage', and 'Nome do sistema de ficheiros' is 'tesedatalakestoragesistemadeficheiros'. At the bottom, there are buttons for 'Rever + criar', '< Anterior', and 'Seguinte: Segurança >'. The 'Rever + criar' button is highlighted in blue.

Figura 17 - Criação da "Área de Trabalho da Synapse" no Azure Synapse

teseareadetrabalho Área de trabalho da Synapse

Procurar (Cmd +/)

Novo conjunto de SQL dedicado + Novo conjunto Apache Spark + Conjunto do Data Explorer (pré-visualização) ...

Descrição geral

Registo de atividade

Controlo de acesso (IAM)

Etiquetas

Diagnosticar e resolver problemas

Definições

Azure Active Directory

Propriedades

Bloqueios

Conjuntos de análise

Conjuntos de SQL

Conjuntos Apache Spark

Conjuntos do Data Explorer (pré-visualização)

Segurança

Encriptação

Rede

Identity

Ligações de pontos finais privados

Inquilinos do Azure AD aprovados

Informações Básicas Vista JSON

Grupo de recursos ([mover](#))
[tese_grupo_recursos](#)

Estado
Succeeded

Localização
East US

Subscrição ([mover](#))
[Azure for Students](#)

ID da Subscrição
fcc22a41-55de-4ad1-a620-79d22cc3d7c3

Rede virtual gerida
Não

ID de objeto da Identidade Gerida
7860e8e5-9adb-48ec-add6-eb193d98f636

URL web da área de trabalho
<https://web.azuresynapse.net/pt-pt/?workspace=%2fsubscriptions...>

Etiquetas ([editar](#))
[Clique aqui para adicionar etiquetas.](#)

Rede
[Mostrar definições da firewall](#)

URL de conta ADLS Gen2 principal
<https://tesedatalakestorage.dfs.core.windows.net>

Sistema de ficheiros ADLS Gen2 principal
tesedatalakestorageisistemadeficheiros

Nome de utilizador de administrador SQL
sqladminuser

Administrador do SQL Active Directory
a86440@uminho.pt

Ponto final de SQL dedicado
teseareadetrabalho.sql.azuresynapse.net

Ponto final de SQL sem servidor
teseareadetrabalho-ondemand.sql.azuresynapse.net

Ponto final de desenvolvimento
<https://teseareadetrabalho.dev.azuresynapse.net>

Introdução

Abrir Synapse Studio
Comece a compilar a sua solução de análise totalmente integrada e desbloqueie novas informações.
[Abrir](#)

Ler documentação
Saiba rapidamente como ser produtivo. Explore conceitos, tutoriais e exemplos.
[Mais informações](#)

Microsoft Azure | teseareadetrabalho

Sinapse em tempo real Validar todos Publicar tudo

Conjuntos de análise

Conjuntos do SQL

Conjuntos Apache Spark

Conjuntos do Data Explorer...

Ligações externas

Serviços associados

Microsoft Purview

Integração

Ativadores

Conjuntos de SQL

O conjunto de SQL sem servidor, Incorporado, fica imediatamente disponível para a sua área de trabalho. É possível configurar Conjuntos de SQL dedicados para restrições organizacionais ou da equipa. [Saiba mais](#)

+ Novo Atualizar

Filtrar por nome

A mostrar 1-2 de 2 itens (1 Sem Servidor, 1 Dedicado)

Nome	Tipo	Estado	Tamanho
Incorporado	Sem servidor	Online	Automático
teseconjuntosqdedicado	Dedicado	Online	DW1000c

Figura 18 - Acesso ao Synapse Studio e criação de um Conjunto SQL dedicado

Anexo C – Carregamento de Dados na Staging Area

Relativamente ao BigQuery, nesta tecnologia não foi necessária a criação das várias tabelas através de código SQL, visto que este tem uma funcionalidade de carregamento de ficheiros CSV do Google Cloud Storage (GCS) para o ambiente do BigQuery, criando automaticamente o esquema das tabelas. Para tal, estes ficheiros CSV foram descarregados das tabelas previamente criadas no Snowflake, obtendo assim um CSV para cada uma das tabelas, com o número de linhas expectado. Para aceder à área do GCS basta pesquisar pelo mesmo na barra de pesquisa e, de seguida, criar um *bucket* novo, onde serão importados os CSVs. Para tal, basta seguir os passos demonstrados na figura 19.

The screenshot shows the Google Cloud Platform interface. At the top, the search bar contains 'cloud storage'. Below the navigation bar, a message states: 'O Cloud Storage oferece um bucket gratuito para cada aplicativo do App Engine, mas você precisa ativar o faturamento se quiser mais armazenamento.' Below this, a table lists the bucket 'tese_tpc_h_2' with details: 'Criado em: 31 de mar. de 2022 16:58:15', 'Tipo de local: Multi-region', 'Local: eu', and 'Default storage class: Standard'. The bucket details page for 'tese_tpc_h_2' is shown, including 'Local: eu (várias regiões na União Europeia)', 'Classe de armazenamento: Standard', 'Acesso público: Não público', and 'Proteção: Nenhum'. Below the details, there are tabs for 'OBJETOS', 'CONFIGURAÇÃO', 'PERMISSÕES', 'PROTEÇÃO', and 'CICLO DE VIDA'. The 'OBJETOS' tab is active, showing a list of CSV files: 'customer.csv', 'lineitem.csv', 'nation.csv', 'orders.csv', 'part.csv', 'partsupp.csv', 'region.csv', and 'supplier.csv'. A red box highlights the 'FAZER UPLOAD DE ARQUIVOS' button.

Nome	Criado em	Tipo de local	Local	Default storage class
tese_tpc_h_2	31 de mar. de 2022 16:58:15	Multi-region	eu	Standard

Local	Classe de armazenamento	Acesso público	Proteção
eu (várias regiões na União Europeia)	Standard	Não público	Nenhum

Nome	Tamanho	Tipo	Criado	Classe de armazenamento	Última modificação	Acesso público
customer.csv	23,3 MB	text/csv	31 de m...	Standard	31 de mar. de 2...	Não público
lineitem.csv	738,6 MB	text/csv	31 de m...	Standard	31 de mar. de 2...	Não público
nation.csv	2,2 KB	text/csv	31 de m...	Standard	31 de mar. de 2...	Não público
orders.csv	163,4 MB	text/csv	31 de m...	Standard	31 de mar. de 2...	Não público
part.csv	22,9 MB	text/csv	31 de m...	Standard	31 de mar. de 2...	Não público
partsupp.csv	113,4 MB	text/csv	31 de m...	Standard	31 de mar. de 2...	Não público
region.csv	415 B	text/csv	31 de m...	Standard	31 de mar. de 2...	Não público
supplier.csv	1,3 MB	text/csv	31 de m...	Standard	31 de mar. de 2...	Não público

Figura 19 - Importação de Ficheiros CSV para o GCS

De seguida, já de volta ao ambiente do BigQuery, deve-se seleccionar o *schema* SA, onde queremos carregar os dados e seleccionar “Criar Tabela”, tal como demonstrado na figura 23. Neste novo painel, representado na figura 20, é necessário escolher a opção de criar uma tabela a partir do GCS, seleccionar o ficheiro CSV do *bucket* previamente criado e seleccionar o formato do arquivo como CSV. Por fim, deve-se definir o nome da tabela e seleccionar “Detetar Automaticamente” na secção “Esquema”, de forma que o esquema da tabela, ou seja, as suas colunas, sejam automaticamente criadas. Este processo deve ser repetido para todos os ficheiros que estão presentes no *bucket* do GCS.

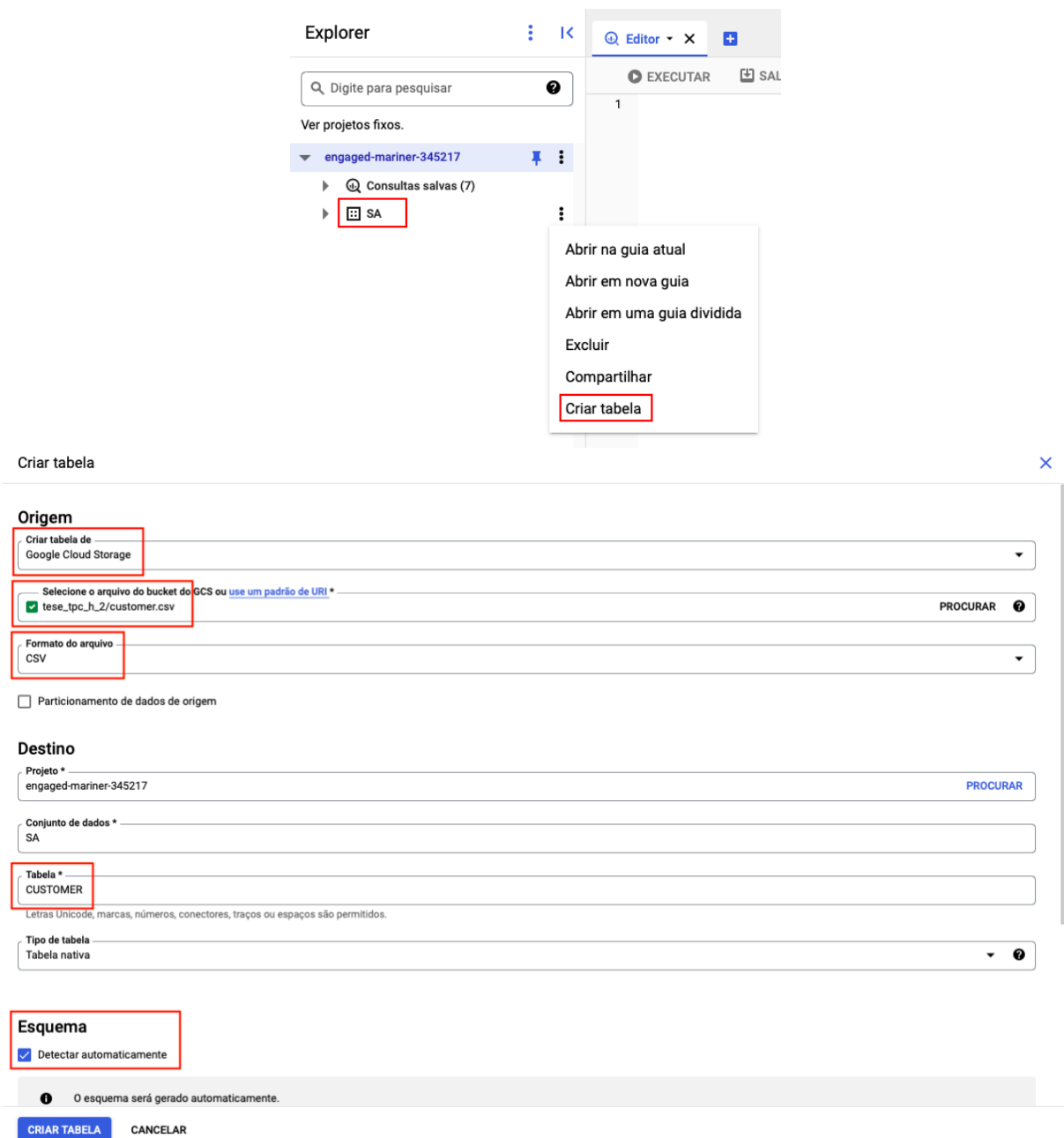


Figura 20 - Criação das Tabelas e Carregamento de dados para a SA no BigQuery

Relativamente ao Azure Synapse, nesta tecnologia também não foi necessária a criação das várias tabelas através de código SQL, visto que este tem uma funcionalidade de carregamento de ficheiros CSV para o DLS e de criação automática de tabelas no conjunto de SQL dedicado, detetando automaticamente o seu esquema. Para tal, utilizando os mesmos ficheiros que foram utilizados no BigQuery, foi realizado este *upload* para o DLS, como está representado na figura 21. Para proceder à criação das tabelas e, conseqüentemente, ao carregamento dos dados, é necessário seguir os procedimentos exemplificados na figura 22, onde é realizado em carregamento em massa.

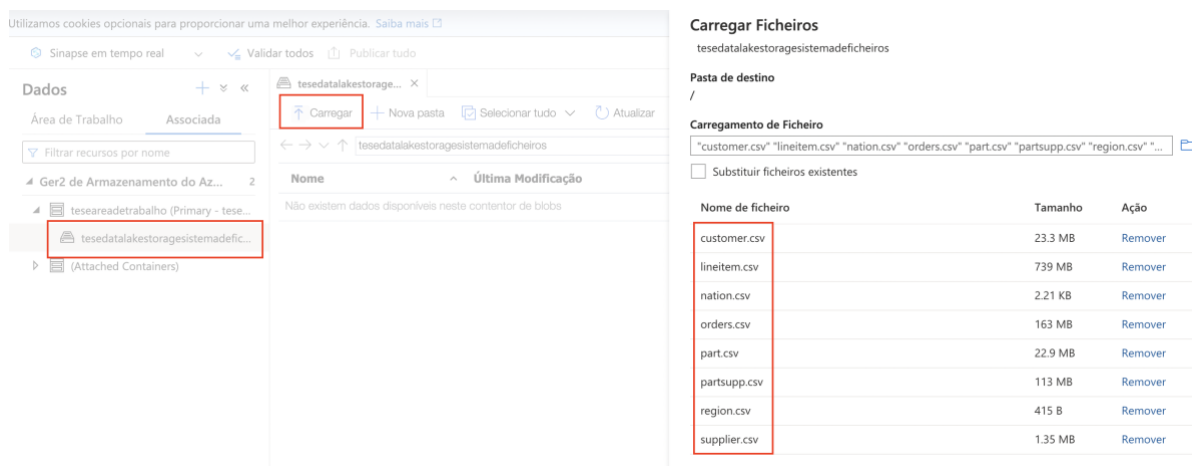


Figura 21 - Upload dos CSV para o DLS no Azure Synapse

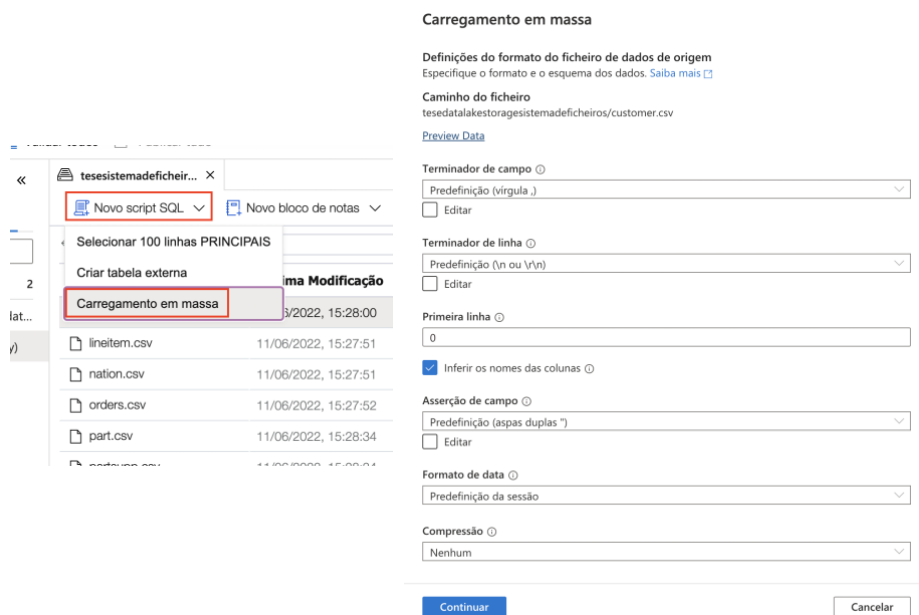
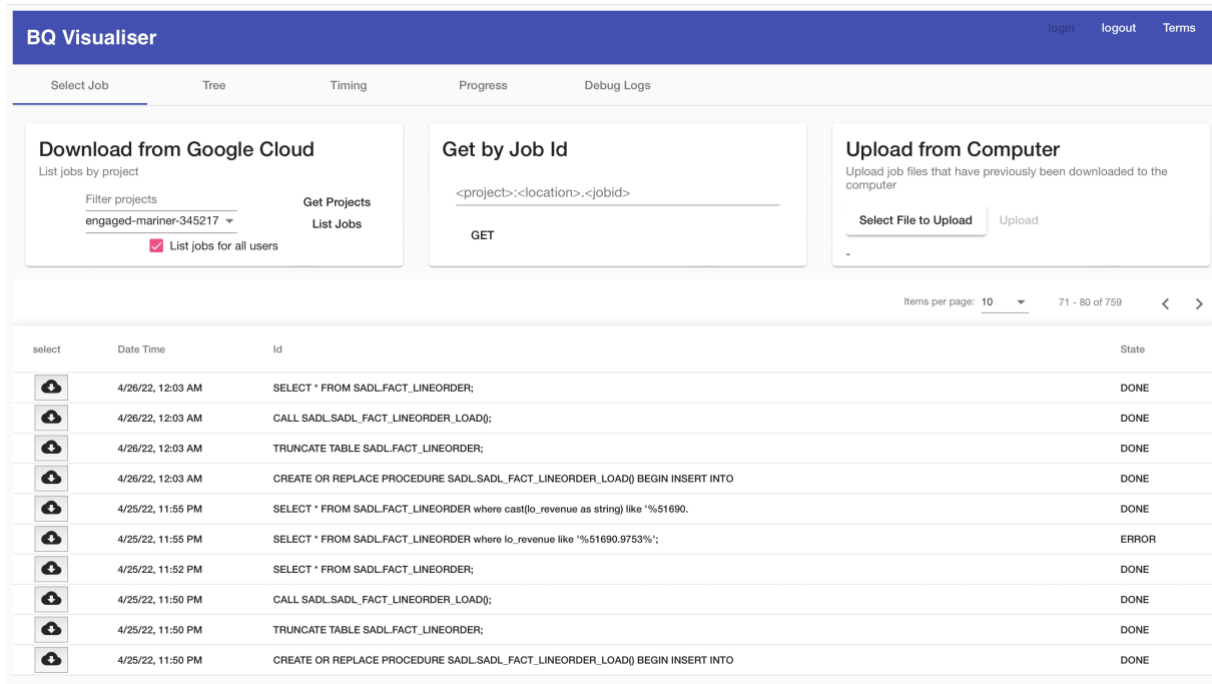


Figura 22 - Criação das tabelas de SA e carregamento dos dados no Azure Synapse

Anexo D – Interface do BQ Visualizer

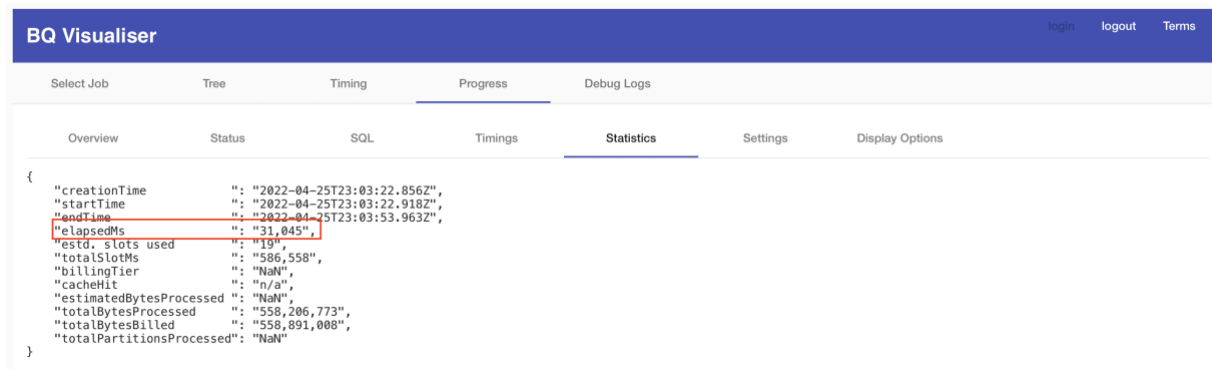
Na figura 23 pode ser visualizada a interface do BQ Visualizer, e na figura 24 está apresentado um exemplo dos resultados apresentados da execução de uma *query*, estando assinalado o resultado que nos importa para esta avaliação.



The screenshot shows the BQ Visualizer interface with the 'Select Job' tab active. It features three main sections: 'Download from Google Cloud', 'Get by Job Id', and 'Upload from Computer'. Below these is a table of jobs with columns for 'select', 'Date Time', 'Id', and 'State'. The table contains 10 rows of job data.

select	Date Time	Id	State
	4/26/22, 12:03 AM	SELECT * FROM SADL_FACT_LINEORDER;	DONE
	4/26/22, 12:03 AM	CALL SADL_SADL_FACT_LINEORDER_LOAD();	DONE
	4/26/22, 12:03 AM	TRUNCATE TABLE SADL_FACT_LINEORDER;	DONE
	4/26/22, 12:03 AM	CREATE OR REPLACE PROCEDURE SADL_SADL_FACT_LINEORDER_LOAD() BEGIN INSERT INTO	DONE
	4/25/22, 11:55 PM	SELECT * FROM SADL_FACT_LINEORDER where cast(lo_revenue as string) like '%51690.	DONE
	4/25/22, 11:55 PM	SELECT * FROM SADL_FACT_LINEORDER where lo_revenue like '%51690.9753%';	ERROR
	4/25/22, 11:52 PM	SELECT * FROM SADL_FACT_LINEORDER;	DONE
	4/25/22, 11:50 PM	CALL SADL_SADL_FACT_LINEORDER_LOAD();	DONE
	4/25/22, 11:50 PM	TRUNCATE TABLE SADL_FACT_LINEORDER;	DONE
	4/25/22, 11:50 PM	CREATE OR REPLACE PROCEDURE SADL_SADL_FACT_LINEORDER_LOAD() BEGIN INSERT INTO	DONE

Figura 23 - Interface do BQ Visualizer



The screenshot shows the BQ Visualizer interface with the 'Progress' tab active. It displays a 'Statistics' section with a JSON object containing various performance metrics. The 'elapsedMs' field is highlighted with a red box.

```
{
  "creationTime": "2022-04-25T23:03:22.856Z",
  "startTime": "2022-04-25T23:03:22.918Z",
  "endTime": "2022-04-25T23:03:53.963Z",
  "elapsedMs": "31,045",
  "estd. slots used": "19",
  "totalSlotMs": "586,558",
  "billingTier": "NaN",
  "cacheHit": "n/a",
  "estimatedBytesProcessed": "NaN",
  "totalBytesProcessed": "558,206,773",
  "totalBytesBilled": "558,891,088",
  "totalPartitionsProcessed": "NaN"
}
```

Figura 24 - Tempo de Execução de Queries no BQ Visualizer