FATEC SP – FACULDADE DE TECNOLOGIA DE SÃO PAULO
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ANDRÉ EIKE TAKEMOTO

BANCO DE DADOS COMO SERVIÇO

SÃO PAULO
2021

FATEC SP – FACULDADE DE TECNOLOGIA DE SÃO PAULO
ANDRÉ EIKE TAKEMOTO

BANCO DE DADOS COMO SERVIÇO

Trabalho submetido como exigência parcial para a obtenção do Grau de Tecnólogo em Análise e Desenvolvimento de Sistemas
Orientador: Me. VALTER YOGUI

SÃO PAULO
2021

FACULDADE DE TECNOLOGIA DE SÃO PAULO

ANDRÉ EIKE TAKEMOTO

BANCO DE DADOS COMO SERVIÇO

Trabalho submetido como exigência parcial para a obtenção do Grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Parecer do Professor Orientador

_____

_____

_____

Conceito/Nota Final: _____

Atesto o conteúdo contido na postagem do ambiente TEAMS pelo aluno e assinada por mim para avaliação do TCC.

Orientador: Me. VALTER YOGUI

SÃO PAULO, ____ de _____ de 2021.

Assinatura do Orientador

Assinatura do aluno 

“Pesquisar é acordar para o mundo”

(Marcelo Lamy)

RESUMO

O armazenamento, recuperação e manipulação de dados é algo primordial para a tomada de decisões, principalmente em uma organização. Dessa forma um projeto mal formulado, que envolve banco de dados pode impactar negativamente nos resultados esperados. Assim, para minimizar qualquer problema futuro é necessário seguir algumas etapas fundamentais como o levantamento de requisitos, elaboração de uma modelagem conceitual do negócio, assim como um modelo lógico e físico. E por último o estabelecimento de regras de segurança. Além dessas etapas, outra questão muito importante é decidir onde serão armazenadas essas informações, ou seja, onde será mantido o servidor do Banco de Dados. Podendo ser *On-Premises* ou em *Cloud*. Caso os gestores do projeto optem por terem total controle sobre o *hardware* e o *software*, ou seja, sobre toda a infraestrutura, configurações, atualizações ou alguma customização, a melhor opção será *On-Premises*. Agora, se é desejado a terceirização dessas responsabilidades, e além disso a escalabilidade automática, a automatização de processos internos e a disponibilidade sob demanda, a hospedagem em *Cloud* torna-se a opção mais viável. Com isso em vista, o presente trabalho tem como finalidade realizar um estudo sobre os conceitos relacionados a Banco de Dados como Serviço. Para isso será apresentada toda a evolução da ferramenta de Banco de Dados até o presente momento, quais são as vantagens e desvantagens perante a solução *On-Premises* e por último uma demonstração prática da utilização da ferramenta.

Palavras-chave: bancos de dados, bancos de dados como serviço, nuvem, *DBaaS*.

ABSTRACT

The storage, retrieval and manipulation of data is essential for decision making, especially in an organization. Thus, a poorly formulated project, which involves a database, can negatively impact the expected results. Thus, to minimize any future problem, it is necessary to follow some fundamental steps such as gathering requirements, elaborating a conceptual business model, as well as a logical and physical model. And finally, the establishment of safety rules. In addition to these steps, another very important issue is to decide where this information will be stored, that is, where the Database server will be kept. It can be On-Premises or Cloud. If project managers choose to have full control over the hardware and software, that is, over the entire infrastructure, configurations, updates or any customization, the best option will be On-Premises. Now, if outsourcing these responsibilities is desired, and in addition to automatic scalability, automation of internal processes and on-demand availability, Cloud hosting becomes the most viable option. With this in mind, this work aims to carry out a study on concepts related to Database as a Service . For this, the entire evolution of the Database tool to date will be presented, what are the advantages and disadvantages compared to the On-Premises solution and finally a practical demonstration of the use of the tool.

Keywords: databases, databases as a service, cloud, DBaaS.

SUMÁRIO

1 INTRODUÇÃO

A tecnologia de banco de dados pode ser considerada como sendo uma das áreas com o crescimento mais rápido dentro da ciência da computação. Seu surgimento ocorreu por volta do final da década de 1960, onde os fabricantes passaram a fornecer *software* que incluíam Banco de Dados (BD).

Com a chegada da computação em Nuvem ocorreu uma das principais mudanças na era da comunicação/informação que trouxe uma arquitetura computacional de fácil acesso e ao mesmo tempo robusta e escalável. Esse paradigma permite uma computação orientada a serviços, abstraindo toda complexidade que um *software* possui e questões de infraestrutura. Em outras palavras, a computação em Nuvem, também conhecida como *Cloud*, fornece serviços de computação de forma mais rápida, com recursos flexíveis e escaláveis.

Esse novo paradigma, que também pode ser conhecido como "*Cloud Computing*" ou então "*as a Service*" possui diversas ofertas de serviços, cada uma com uma finalidade diferente e específica para cada necessidade do cliente.

O ecossistema de computação em Nuvem tradicional que conhecemos é dividido em basicamente três produtos. O mais popular é o *Software as a Service (*SaaS*)*, seguido do *Plataform as a Service* (PaaS) e por último *Infrastructure as a Service (IaaS)*. Seguindo essa tendência, com Banco de Dados não poderia ser diferente. Dessa forma esse paradigma se estende também na ciência de dados como *DataBase as a Service (DBaaS)*.

2 OBJETIVOS

Fazer uma abordagem conceitual a respeito da tecnologia de Banco de Dados como Serviço de uma forma simplificada e objetiva, para que ao final seja apresentada uma utilização prática de uma plataforma que provê o serviço.

Para isso é necessário apresentar o conceito da tecnologia de Banco de Dados e a sua evolução até o presente momento e posteriormente, após ter estabelecido o princípio fundamental da tecnologia, será explanada a tecnologia de Computação em Nuvem, seguida da tecnologia de Banco de Dados como Serviço.

Dessa forma almeja-se que quaisquer tipo de dúvida ou curiosidade a respeito da tecnologia de Banco de Dados como Serviço seja esclarecida através desse trabalho, visto que apesar de ser relativamente nova e talvez não muito difundida, essa tecnologia abstrai diversos processos com uma razoável simplicidade técnica.

3 EVOLUÇÃO DA TECNOLOGIA DE BANCO DE DADOS

3.1 Introdução

Antes do surgimento dos Bancos de Dados, os dados eram armazenados em arquivos físicos e organizados em pastas. Extrair e gerenciar as informações nesses arquivos era uma tarefa que podia durar horas, sem contar que cada fonte de informação não necessariamente estavam na mesma localidade geográfica. A redundância e inconsistência dos dados era outro problema que podia ocorrer, devido ao fato de uma informação poder estar presente em mais de um documento e, além disso, ser diferente.

Tendo em vista todos esses problemas em gerenciar dados, surgiu a necessidade de aprimorar a forma como os dados eram acessados e manipulados, uma vez que estava claro que era preciso um meio mais eficiente de armazenar e manter os dados para posterior recuperação. Isso veio a ser resolvido com o surgimento dos computadores pessoais e os Sistemas de Gerenciamento de Banco de Dados (SGBD).

Neste capítulo será apresentado como foi o surgimento dos primeiros Bancos de Dados e a evolução dessa tecnologia até os dias atuais. Dessa forma será possível compreender os próximos capítulos.

3.2 Década de 1960

A empresa de tecnologia *International Business Machines Corporation* (IBM) foi uma das responsáveis pelo surgimento das primeiras ferramentas de armazenamento de dados, visando o aprimoramento de processos de manipulação e armazenamento de arquivos em escritórios. Nessa época as empresas descobriram que os custos relacionados com o pagamento de funcionários somente para o gerenciamento de arquivos estavam começando a ficar elevados. Dessa forma, foi necessário o investimento em tecnologia para tornar os processos mais eficientes e menos custosos.

Como nessa época os dados eram armazenados diretamente em arquivos físicos haviam inúmeros problemas relacionados com dificuldade de acesso, redundância, inconsistência e segurança (BARCELAR, 2012, p.3).

Com a adoção dos computadores por parte das empresas, houve um grande crescimento na capacidade de armazenamento, fazendo com que a tecnologia passasse a fazer parte dos custos efetivos das empresas. Nessa época. Foram desenvolvidos dois principais modelos de dados: modelo em rede *Conference on Data Systems Languages* (CODASYL) e o *Information Management System* (IMS) (BERG; SEYMOUR; GOEL, 2012, p.30).

Segundo Berg e Seymour (2012, p.30), como o acesso ao Banco de Dados era realizado através de ponteiros de baixo nível, os detalhes de armazenamento dependiam do tipo de informação que fosse ser armazenada, assim adicionar um simples campo em uma tabela poderia ocasionar em uma reescrita da estruturação do modelo de dados. Dessa forma os usuários precisavam conhecer a estrutura física do BD para poder realizar uma simples consulta.

3.3 Década de 1970

Em Junho de 1970 o pesquisador da IBM Dr. Edgar Frank Codd teve um artigo publicado na revista “Comunicações da ACM” intitulado como “um modelo relacional para grandes bancos de dados”. Com isso o modelo proposto por Codd passa a ser aceito como modelo definitivo para SGBD.

Codd diz em seu artigo que a maneira como os dados são armazenados nas máquinas devem ser abstraídos para os usuários. E suas atividades não devem ser prejudicadas caso haja alguma modificação na representação interna ou externa dos dados. Visto que tais mudanças são necessárias, como resultado de alterações no tráfego de consulta, atualização e extração de relatório, que são naturais conforme o crescimento das informações armazenadas (CODD, 1970).

Como esse artigo trata sobre a utilização de cálculo e álgebra relacional para que assim usuários sem conhecimentos técnicos pudessem manipular grande quantidade de informação,

Codd tinha em mente um sistema que armazena informações em tabelas e que também fosse possível que um usuário comum acessasse informações através de simples comandos (BARCELAR, 2012, p.4).

Ainda segundo Barcelar (2012, p.4), a *IBM* criou um grupo de pesquisa conhecido como *System R*, que deu origem a um Sistema de Gerenciamento de

Banco de Dados Relacional (SGBDR) com o mesmo nome. O *Sistema R* evoluiu para SQL/DS, que posteriormente passou a chamar o *DB2*, o mais conhecido sistema gerenciador de Banco de Dados da *IBM*. O *Systema R* utilizava uma linguagem chamada *Structured Query Language* (SQL). Linguagem esta que se tornou um padrão na indústria para Bancos de Dados relacionais e hoje em dia é um padrão *International Organization for Standardization* (ISO).

Apesar de muitos considerarem o *DB2* da *IBM* como o primeiro SGBD, segundo Kriegel (2011, p.6) essa afirmação é contestada por outros autores que apontam o sistema comercial da Oracle lançado em 1979 como sendo o primeiro. Já outros pesquisadores, o *Multics Relational Data Store* vendido pela *Honeywell Information Systems* em 1976. Ou os projetos experimentais *Micro DBMS* (pioneiros em alguns dos princípios formulados pelo Dr. Codd dois anos depois) da Universidade de Michigan de 1968.

Outro colaborador importante que expandiu os conceitos existentes sobre modelagem de dados foi o Dr. Peter Chen ao apresentar o modelo Entidade-Relacionamento. Este modelo incorpora algumas das informações semânticas importantes sobre o mundo real e pode ser usado como base para a unificação de diferentes visões de dados como o modelo de rede, o modelo relacional e o modelo de conjunto de entidades (CHEN, 1976, p.1)

3.4 Década de 1980

Em 12 de agosto de 1981, em uma conferência de imprensa em Nova York, Philip Donald Estridge anunciou o primeiro computador de uso pessoal, o *IBM Personal Computer* custando US $1.565,00 (IBM, 2003). Nessa época a venda em larga escala de computadores inicia a comercialização de sistemas relacionais, tornando o *SQL* um padrão mundial. Dessa forma o *DB2* torna-se o principal produto da IBM e os modelos hierárquico e de rede passam a não fazerem mais tanto sucesso, porém ainda sendo utilizados em sistemas legado. (BERG; SEYMOUR; GOEL, 2012, p.32)

Com o surgimento de novas necessidades ficou claro que em algumas áreas o modelo relacional não era aplicável, devido aos tipos de dados envolvidos, por exemplo: medicina, área que precisa de flexibilidade na forma como os dados são representados e acessados (BARCELAR, 2012, p.5).

Outro exemplo são os sistemas computacionais *Computer Aided Design* (CAD) e *Computer Aided Manufacturing* (CAM) utilizados para prototipagem e manufatura. Ferramentas *Computer-Aided Software Engineering* (CASE) utilizadas para análise de requisitos, modelagem, programação e testes. E *Geographic Information System* (GIS) utilizados para análise e gestão de um espaço geográfico.

Com a popularidade das linguagens de programação Orientadas a Objetos como *C++, Java, C#, Visual Basic.Net, Delphi, Smalltalk* e *Eiffel* surgiu também a ideia de armazenamento de informação em objetos, visto que apesar eficácia dos Bancos de Dados no modelo relacional, havia uma necessidade de atender aos novos tipos de dados que foram surgindo (KRIEGEL, 2011, p.348).

> (...) Imagine uma classe Java LIVRO, que tem suas propriedades (atributos) e métodos definidos. Quando um aplicativo cria uma instância desta classe para adicionar um novo livro, preenche suas propriedades e salva a instância em um banco de dados. No contexto SGBDR, isso significaria extrair dados e preencher várias tabelas. Para um banco de dados OO, significa serializar o objeto como ele é - como um binário, código de byte ou versão de descrição de texto - e ainda ser capaz de rastrear este objeto por, digamos, PK_ID, sem a necessidade de reunir todos os dados relevantes da tabela. É disso que tratam os SGBDOO e SGBDOR. (KRIEGEL, 2011, p.348, tradução).

Com o crescente desenvolvimento de sistemas em diversas áreas e com o aumento do uso de computadores pessoais, foram surgindo novas versões de *software* de Banco de Dados com correções feitas através do *feedback* dos usuários, sendo a *Exodus* e a *Orion* em 1986 e a O2 em 1988, a implementarem o padrão orientado a objetos. Já a *Postgres* e a *Starburst* mesclaram características da orientação a objeto com o modelo relacional visando consultas mais performáticas. E foi somente no final da década que a tecnologia dos sistemas relacionais passaram a apresentar uma maturidade considerável, como a *DB2*, a *Ingres*, a *Oracle*, a *Sybase* e a *Informix* (BARCELAR, 2012, p.6).

3.5 Década de 1990

Esse período foi marcado pela diminuição das empresas de tecnologia e a estabilização das existentes, que passaram a oferecer produtos mais sofisticados e com um preço mais elevado. Os principais produtos desenvolvidos nesse período foram ferramentas de desenvolvimento de *software*, como o *Visual Basic* (VB) da *Microsoft*, o *Oracle Developer* e o *PowerBuilder* da *Sybase*. Surgiram ferramentas de produtividade pessoal como o *Open DataBase Connectivity*, o *Excel* e o *Access* da *Microsoft*. E alguns dos produtos desenvolvidos nesse período apresentavam tecnologia *real-time*, além de avanços na padronização de interfaces e interoperabilidade.

O principal marco desse período foi a comercialização e a popularização da internet, e com isso o modelo cliente-servidor tornou-se uma referência para desenvolvimento de *software*, permitindo a distribuição de tarefas entre os fornecedores e requerentes do serviço.

Ferramentas *World Wide Web* (Web) como *FrontPage*, *Java Servlets*, *Java Database Connectivity* (JDBC), *Enterprise Java Beans*, *ColdFusion*, *Dream Weaver* e *Oracle Developer 2000* são exemplos de produtos desenvolvidos também nesse período, juntamente com soluções *open source* como o *GNU Compiler Collection* (GCC), *Computer Generated Imagery* (CGI), *Apache* e *MySQL* que foram amplamente utilizados. O Processamento de Transações Online e o Processamento Analítico Online atingiram a maturidade com muitos comerciantes usando a tecnologia de ponto de venda diariamente (BERG; SEYMOUR; GOEL, 2012, p.32).

Em 1997 surgiu o *Extensible Markup Language* (XML), uma linguagem de marcação que define um conjunto de regras para a codificação de documentos em um formato tanto legível por humanos quanto por máquinas. Os objetivos de *design* do XML enfatizam a simplicidade, generalidade e usabilidade na *Internet*. É um formato de dados textuais com forte suporte via *Unicode* para os idiomas do mundo. Embora o *design* de XML se concentre em documentos, ele é amplamente usado para a representação de estruturas de dados arbitrárias, como serviços da Web. Muitas Interfaces de Programação de Aplicativos foram criadas para desenvolvedores de *software* usarem para processar os dados XML, e vários sistemas de esquema XML existem para ajudar na definição de linguagens baseadas em XML. O XML se aplicava ao processamento de Banco de Dados, o que resolvia problemas antigos de Banco

de Dados. Os principais fornecedores começam a integrar XML em produtos SGBD (BERG; SEYMOUR; GOEL, 2012, p.32).

3.6 Década de 2000

Muitas mudanças tecnológicas ocorreram nesse período, resultantes do ritmo acelerado de desenvolvimento e aprimoramento de *software* decorrentes do receio do *Bug do Milênio*. Entretanto, houve um declínio acentuado da indústria da *Internet*, apesar de que aplicativos de Banco de Dados continuaram em crescimento, sendo as empresas *IBM*, *Microsoft* e *Oracle* a dominam esse mercado.

Alguns Bancos de Dados passaram a apresentar um certo grau de lógica de programação, como disparar para o usuário uma mensagem de alerta sobre algum problema ocorrido, devido a inconsistência no código executado; recurso de gatilho para desencadear uma série de procedimentos; atualização em cascada, para evitar inconsistência entre entidades no Banco de Dados, após a mudança de estado de algum registro; estrutura de *loop*, para iteração sobre algum recurso. Alguns desses produtos são *Sybase/SQL Servers*, *Oracle's PL/SQL* e *PostgreSQL* (BERG; SEYMOUR; GOEL, 2012, p.33).

Embora o termo *NoSQL* tenha surgido em 1998, por Carlo Strozzi, foi somente em 2009 que o termo ganhou fama, visto que nesse período havia uma grande necessidade de consultas performáticas, diminuição de custos relacionados a armazenamento e infraestrutura (KRIEGEL, 2011, p.331). O termo *NoSQL* originalmente se referia a "não SQL", no sentido de não haver a necessidade de relacionamento entre entidades em um Banco de Dados, que posteriormente se estendeu para "não somente *SQL*", trazendo um significado mais flexível (CUPPETT, 2016, p.52).

Segundo Kriegel, os Bancos de Dados *NoSQL* são categorizados de acordo com a maneira como armazenam os dados e se enquadram em categorias como armazenamentos de chave-valor, implementação *BigTable* (Google), Bancos de Dados para armazenamento de documentos e Bancos de Dados Gráficos. Seguindo esse raciocínio, Berg e Seymour complementam dizendo que os sistemas de Banco de Dados *NoSQL* tiveram um grande desenvolvimento ao lado de grandes empresas como *Google*, *Amazon*, *Twitter* e o *Facebook*, que tinha desafios significativamente

diferentes ao lidar com dados que as soluções de SGBDR tradicionais não conseguiam lidar. (BERG; SEYMOUR; GOEL, 2012, p.6).

Alguns dos principais Bancos de Dados não relacionais são o *MongoDb* que possui uma estrutura de dados semelhante ao formato *JavaScript Object Notation* (JSON) para armazenamento de dados; *Cassandra*, originalmente foi desenvolvido pelo *Facebook*, que possui um Banco de Dados Orientado a Colunas, permitindo consultas mais performáticas; *Redis* que tem sua implementação baseada no formato chave-valor permitindo que o acesso aos dados seja realizado de forma mais performática; *HBase* é uma implementação do *BigTable* (*Google*), também possui um Banco de Dados Orientado a Colunas e possui implementações de empresas como *LinkedIn*, *Facebook* e *Spotify*; *Neo4j* que relaciona os dados baseado no sistema de grafos (*Graph Database*) e se destaca em alguns cenários como o de mineração de dados e reconhecimento de padrões; *Amazon DynamoDb* é um serviço de Banco de Dados não relacional da plataforma *Amazon Web Services* (AWS), compatível com o padrão chave-valor e possui baixa latência, aproximadamente 10 milissegundos em qualquer escala (MATOS, 2018).

3.7 Atualmente

Nota-se uma tendência cada vez mais acentuada de um distanciamento entre o computador e a máquina individual. Inicialmente, os dados eram armazenados em servidores para que assim os computadores clientes pudessem ter acesso a uma informação, e posteriormente as conexões por cabos foram substituídas por aquelas fornecidas como parte da infraestrutura da *Internet*. E agora, no cenário atual, os servidores locais podem ser substituídos por serviços em Nuvem. Não é mais necessário comprar *hardware* e *software*, e ter uma máquina servidor dedicada, basta adquirir um servidor virtual de um provedor de serviços conforme a necessidade.

Segundo a empresa de consultoria *Gartner*, é previsto que 75% de todos os Bancos de Dados serão implementados ou migrados para uma plataforma baseada em serviço de computação em Nuvem até o ano de 2022. Além disso, a pesquisa mostra também que 68% de todo o investimento em Sistemas de Gerenciamento de Banco de Dados são destinados para o setor de *Cloud DataBase*. Dessa forma empresas como a *Microsoft* e a *Amazon Web Services* se estabelecem como as mais

requisitadas, correspondendo a 75,5% de todo o crescimento do mercado (REDATOR, 2019).

Em outra pesquisa, a mesma empresa de consultoria diz que no ano de 2023 a receita de Sistemas de Gerenciamento de Banco de Dados baseados em Nuvem será responsável por 50% da receita total desse mercado e que a preferência da Nuvem para gerenciamento de dados reduzirá o cenário do fornecedor, enquanto que o crescimento da *multicloud* aumentará a complexidade para governança e integração de dados (GARTNER, 2020).

Como vimos, os dados estão cada vez mais se movendo para a Nuvem. E tarefas como manipulação e obtenção de grande quantidade de dados, que podem consumir muito recurso computacional, podem ser facilmente executadas em Nuvem à medida que tais recursos serão disponibilizados de forma indefinida conforme a necessidade para processamento. Essa disponibilidade traz diversas vantagens para a empresa e os programadores, uma vez que questões relacionadas à infraestrutura e manutenção passam a ser responsabilidade do provedor do serviço, deixando os programadores responsáveis somente por desenvolverem a aplicação.

Como o armazenamento de dados está se tornando cada vez mais complexo, os dados requerem a necessidade de funções especializadas. Assim, segundo Juan Loaiza, outras tendências para banco de dados são: implementações específicas para atenderem aos dispositivos *Internet of Things* (IoT); integração entre Banco de Dados e *Blockchain*, para análise de informação da rede; Computação em Hiperescala onde os dados são fragmentados em diversos Bancos de Dados para facilitar o processamento de dados com o auxílio da tecnologia *NoSQL* (REICKSON, 2019).

4 BANCO DE DADOS

4.1 Introdução

Após conhecer como surgiu e como foi a evolução que a tecnologia de Banco de Dados teve até o presente momento, é necessário apresentar alguns conceitos básicos sobre Banco de Dados para o entendimento dos próximos capítulos, visto que existem diversas variedades dessa ferramenta, podendo ser desde milhões de discos rígidos armazenados em uma sala gerenciados por um computador ou até uma simples planilha de *Excel*. Esses dados geralmente são armazenados em uma estrutura que pode ser representada por algo semelhante a uma simples planilha, porém há um sistema que é responsável pelo gerenciamento, extração, integridade e manipulação desses dados.

4.2 O que é Banco de Dados?

Banco de Dados é uma coleção organizada e estruturada de informações ou dados relacionados que geralmente são armazenados de forma eletrônica em um computador (ORACLE, 2014). Outras definições, como a de Barcelar, define Banco de Dados como um conjunto de informações com uma estrutura regular, normalmente, mas não necessariamente, armazenadas em algum formato de máquina legível para um computador (BARCELAR, 2012, p.9).

Para gerenciar esses dados é necessário um Sistema de Gerenciamento de Banco de Dados, ou seja, um *software* capaz de prover tais recursos. O gerenciamento abrange tarefas como a definição da estrutura que armazenará as informações, através de um conjunto de tabelas relacionadas, e mecanismos de manipulação de dados. Outra característica de um SGBD é a definição de regras de segurança, dessa forma um determinado usuário poderá ou não ter acesso a informações contidas em uma tabela ou executar determinados comandos.

4.3 Dado

Dado pode ser definido como uma sequência de símbolos quantificados ou quantificáveis. Exemplificando, um texto pode ser definido como um dado, pois as letras contidas nos textos são símbolos quantificados. Dessa forma, imagens, sons e

animações também são dados, pois todos podem ser quantificados de uma forma que qualquer pessoa que tiver contato com algum desses, possivelmente encontrará dificuldades para distinguir a sua reprodução, baseado na representação quantificada, com o original. E como são símbolos quantificáveis, dados podem ser armazenados em um computador e processados por ele. Assim, um dado é necessariamente uma entidade matemática. Isto significa que os dados podem ser totalmente descritos através de representações formais e estruturais. Dentro de um computador, trechos de um texto podem ser ligados virtualmente a outros trechos, por meio de proximidade física ou por ponteiros, ou seja, endereços da unidade de armazenamento sendo utilizada (SETZER, 1999).

Como definido anteriormente, um Banco de Dados é basicamente um depósito de dados armazenados. Segundo Date, tais dados podem ser tanto compartilhados quanto integrados. A integração, nesse contexto, tem por definição a unificação dos dados distintos contidos em um arquivo, dessa forma a redundância que poderia existir é eliminada parcialmente ou até totalmente. Por exemplo, em um processo de administração de cursos, onde é necessário saber a informação de qual departamento cada estudante está matriculado e os arquivos existentes para pesquisa são os registros de Empregado, contendo as informações de nome, endereço, telefone, salário e departamento, e os registros de Matrícula, contendo as informações de matrícula de empregados em curso de treinamento. Nesse caso, o registro de Matrícula é desnecessário, visto que tal informação já contém nos registros de Empregado. Já o compartilhamento pode ser descrito como uma consequência do fato do Banco de Dados ser integrado, em outras palavras, os dados acessados por um usuário, pode estar sendo acessado por diversos outros usuários ao mesmo tempo. E um Banco de Dados capaz de suportar esse compartilhamento também pode ser definido como sistema de múltiplos usuários (DATE, 1989, p.27).

4.3.1 Tipos de Dados

Em um Banco de Dados, cada coluna representa uma informação e possui um tipo de dado relacionado. O tipo de dado é um atributo que especifica o tipo de dado que o objeto pode manter, por exemplo números inteiros, caracteres, valores monetários, data, hora, cadeias de caracteres e binárias.

4.4 Modelo de Dados

Para garantir a integridade dos dados e o bom desempenho proporcionado por um Banco de Dados é importante que o programador que desenvolve a aplicação tenha conhecimento de qual modelagem de dados irá utilizar como base, pois somente dessa forma será garantido que qualquer problema relacionado a inconsistência de dados será minimizada ou até inexistente.

Assim, a modelagem de dados pode ser definida como sendo um conjunto de conceitos utilizados que descrevem a estrutura lógica e física de um Banco de Dados (BARCELAR, 2012, p.19). Quanto mais próximo a modelagem de dados for do ambiente proposto, ou também conhecido como mini mundo, maiores serão as chances do projeto apresentar o retorno esperado.

Por outro lado, um projeto que não possui uma modelagem de dados bem definida e que não está próxima do ambiente de análise, pode resultar em grandes problemas para a organização, comprometendo o sucesso do projeto, pois a aplicação passará a apresentar dados inconsistentes e baixo desempenho, e com isso, diminuindo a confiança dos gestores no projeto.

Em suma, a modelagem é basicamente a criação de uma abstração da realidade de uma forma que seja possível a compreensão e o armazenamento de tal informação. E por possuir uma notação em idioma natural, possibilita que usuários comuns compreendam o modelo desenvolvido, sendo possível a sua colaboração na etapa de revisão.

4.4.1 Modelo Conceitual

O Modelo Conceitual é a descrição de mais alto nível de um Banco de Dados, ou seja, é uma descrição mais abstrata da realidade, pois não contém detalhes técnicos de implementação. Esse modelo é utilizado para que desenvolvedores e usuários possam se comunicar a respeito do projeto, e isso é possível devido a sua facilidade de compreensão por parte dos usuários comuns.

Segundo Barcelar (2012, p.20), o Modelo Conceitual deve ser sempre a primeira etapa de um projeto de um Banco de Dados, visto que seu objetivo é representar a semântica da informação, independente de considerações de eficiência, para que os usuários finais tenham entendimento do contexto a ser desenvolvido. E

nessa etapa é definida a realidade da organização, sendo estabelecidas as regras de negócio, nível de acesso, departamentos e etc.

É importante ressaltar que o Modelo Conceitual não está relacionado com o Modelo de Banco de Dados, forma de acesso ou armazenamento dos dados. Ele está focado em uma representação gráfica de uma realidade existente em um contexto de negócio.

4.4.2 Modelo Lógico

A realização do Modelo Lógico é desenvolvido após a concretização do Modelo Conceitual e nessa etapa é feita a representação das estruturas que irão armazenar os dados, ou seja é definida quais serão as tabelas que irão compor o Banco de Dados, assim como as propriedades e os relacionamentos

O foco do Modelo Lógico é em minimizar ao máximo a quantidade de tabelas que existirão no Banco de Dados. E também é considerado o Modelo de Banco de Dados que será utilizado.

4.4.3 Modelo Físico

O modelo Físico é baseado no Modelo Lógico definido. Nessa etapa é considerada questões técnicas relacionadas ao tipo e tamanho do campo que armazenará os dados, relacionamento entre tabelas, indexação de campos, restrições, gatilhos, funções, visões e procedimentos.

Nessa etapa também é definida qual linguagem será utilizada para a construção do Banco de Dados.

4.5 Modelo de Banco de Dados

O Modelo de Banco de Dados é um conjunto de ferramentas que possibilitam demonstrar como as estruturas de dados serão construídas.

Conforme apresentado no primeiro capítulo, os primeiros modelos a serem desenvolvidos foram o Modelo Hierárquico e o de Rede. E posteriormente, com o surgimento de novas necessidades, surgiu o Modelo Relacional, que em pouco tempo

foi aderido pelas organizações, visto que possuía uma estrutura extremamente simples e solucionava os problemas dos modelos anteriores.

4.5.1 Modelo Hierárquico

O Modelo de Hierarquia representa os dados de cima para baixo, semelhante a uma estrutura de árvore invertida, onde o primeiro registro no topo da árvore é usualmente conhecido como “raiz”. De forma geral, a raiz pode ter qualquer quantidade de registros dependentes a ela, que por sua vez, também podem ter qualquer quantidade de registros dependentes em níveis mais baixos, e assim sucessivamente (DATE, 1989, p.80).

É normal dentro do Modelo Hierárquico que qualquer ocorrência de registro só apresente seu verdadeiro significado, quando ele é visto dentro de um contexto, até porque nenhuma ocorrência dependente de registro pode existir sem a existência de um registro superior. Dessa forma, ao realizar consultas nesse modelo, é necessário adicionar o comando que identifica qual é o registro superior daquela ocorrência.

Embora o Modelo Hierárquico apresente um nível de complexidade a partir do momento que o Banco de Dados vai crescendo, essa abordagem segue um caminho semelhante ao que observamos no mundo real.

Esse modelo tende a ser complexo para os usuários devido a perda de simetria das informações, pois alguns registros passam a ser dependentes de outros, mesmo esse não sendo verdadeiramente dependente.

4.5.2 Modelo de Rede

O Modelo de Rede é uma extensão do Modelo Hierárquico, sendo os dados representados por registros, interligações e conectores. O que diferencia os dois modelos é o fato de ser possível ,no Modelo de Rede, ter qualquer quantidade de registros superiores, não sendo limitado somente a um registro, como é o caso do Modelo Hierárquico (DATE, 1989, p.82).

O tipo de registro conector, nesse modelo, representa a associação entre duas entidades. Para exemplificar podemos imaginar um cenário onde existe uma entidade chamada Comprador, uma chamada Fornecedor e outra chamada Peça. O conector

neste relacionamento entre essas entidades representa a quantidade de Peças que um Fornecedor vende para um Comprador.

Nesse modelo a relação entre os registros é do tipo membro-proprietário, na qual um membro pode ter muitos proprietários. Assim é possível que exista mais de um caminho pelo qual um determinado elemento de dados pode ser acessado (BARCELAR, 2012, p.10).

4.5.3 Modelo Relacional

No Modelo Relacional os dados são representados em tabelas bidimensionais. Cada tabela em um Banco de Dados é na realidade um tipo especial de construção, que na matemática é conhecida como relação, termo que tem uma definição muito mais precisa do que os tradicionais que conhecemos: tabela, e em alguns casos, arquivo (DATE, 1989, p.77).

Nessa abordagem os dados estão baseados na observação de que tabelas que seguem uma determinada limitação pode ser considerada uma relação matemática e dessa forma, em conjunto com a teoria elementar de relações, podem ser utilizadas para resolver problemas práticos relacionados à obtenção de dados em arquivos.

Cada linha em uma tabela é chamada de tupla, que representa um único registro, e esse termo também pode ser definido como par. E cada coluna é usualmente conhecido como atributo.

Esse modelo se tornou um dos mais populares, pois atendia às necessidades dos usuários de Banco de Dados e com baixa complexidade de usabilidade, pois para obter e manipular os dados existentes, basta utilizar operações de álgebra relacional. E o gerenciamento dos dados inclui comandos de seleção, projeção, agrupamento, deleção e alteração.

4.6 SQL

Antes do surgimento dos Bancos de Dados de forma comercial, os desenvolvedores tinham que implementar seus próprios sistemas de armazenamento, caso desejassem o armazenamento de dados persistente e geralmente os dados eram mantidos em arquivos de texto ou binário, sendo restrito somente para leitura e inserção de dados. Isso fazia com que todos os sistemas de armazenamento de dados

fossem estritamente compatíveis com a estrutura dos arquivos que continham os dados, assim qualquer alteração no sistema ou no arquivo resultaram em grandes mudanças para compatibilidade. Além disso, cada sistema tinha os seus próprios métodos de acesso, dessa forma cada fornecedor tinha uma mecanismo específico para acessar os dados e isso agravava ainda mais a complexidade.

Por mais que o Modelo Relacional tenha solucionado alguns problemas da época, questões como as especificações de armazenamento e recuperação de dados não eram abordadas, somente especificava que o modelo deveria ser baseado em conjuntos de relações e seguir as regras de álgebra relacional para extrair informações. Com isso, os primeiros SGBD implementaram várias linguagens diferentes como o *Structured English Query Language* (SEQUEL*)* e *Query Language* (*QUEL)* que acabaram resultando no SQL.

A linguagem SQL foi projetada para definir estruturas relacionais em um SGBD e fornecer recursos para gerenciamento de dados, como recuperação, alteração, deleção e inserção.

A SQL é uma linguagem de programação projetada para definir construções relacionais (como esquemas e tabelas) e fornecer recursos de manipulação de dados, ou seja, ela instrui o Banco de Dados sobre o que você deseja fazer e deixa os detalhes de implementação para o próprio SGBD.

4.6.1 Padrão SQL

Na década de 1980, o *American National Standards Institute* (ANSI) começou a trabalhar no padrão SQL e o objetivo foi o de introduzir um padrão que permitisse a portabilidade entre sistemas de armazenamento de dados.

Desde a sua primeira versão (SQL-86), em 1986, foram implementadas nove versões do padrão SQL ANSI, sendo a versão de 2019 (SQL:2019) a última, até o presente momento (KOZUBEK, 2020).

4.6.2 Dialetos de SQL

Apesar de existir o padrão SQL, existem diversas variações desenvolvidas por fornecedores de *software*, seja para atender as necessidades da comunidade de usuários já existentes antes do padrão protocolado pela ANSI ou por questões de competitividade entre fornecedores.

Alguns dialetos populares de SQL incluem os seguintes:

- PL/SQL - encontrado no Oracle PL/SQL, que significa Linguagem de Procedimentos/SQL e contém muitas semelhanças com a linguagem de programação *Ada*.
- Transact-SQL - usado tanto pelo *Microsoft SQL Server* quanto pelo *Sybase Adaptive Server*.
- SQL PL - extensão procedural do IBM DB2 para SQL.
- PL/pgSQL - implementadas no *PostgreSQL*, que significa Linguagem Procedural/PostgreSQL.
- MySQL - o *MySQL* introduziu uma linguagem procedural em seu banco de dados na versão 5, mas não há um nome oficial para ele.

4.6.3 Operadores SQL

Kriegel diz que apesar do padrão SQL ser conhecido como uma linguagem de consulta de dados, o SQL possui recursos para definição de estrutura de dados, como inserção, deleção e alteração. Também especifica restrições para integridade dos dados e mais outros recursos, que serão abordados a seguir (KRIEGEL, 2011, p.47).

4.6.3.1 Linguagem de Definição de Dados

Linguagem de Definição de Dados ou *Data Definition Language* (DDL) pertence a um grupo de operadores de definição de dados, ou seja, com o auxílio dos operadores pertencentes a esse grupo, podemos definir a estrutura e gerenciar os objetos desta base de dados.

Os operadores pertencentes ao grupo DDL são:

- *Create* - cria objetos no banco de dados;
- *Alter* - modifica objeto existentes no banco de dados;
- *Drop* - excluir objetos no banco de dados.

4.6.3.2 Linguagem de Manipulação de Dados

Linguagem de Manipulação de Dados ou *Data Manipulation Language* (DML) pertence a um grupo de operadores para manipulação de dados. Com esses

operadores podemos extrair, inserir, deletar e atualizar dados do Banco de Dados, ou seja, manipulá-los.

Os operadores pertencentes ao grupo DML são:

- *Select* - extrai de dados de um determinado objeto;
- *Insert* - adiciona novos registros em um determinado objeto;
- *Update* - altera dados de um registro existente;
- *Delete* - deleção de dados.

4.6.3.3 Linguagem de Controle de Dados

Linguagem de Controle de Dados ou *Data Control Language* (DCL) pertence a um grupo de operadores que define o acesso aos dados. Em outras palavras, são operadores de gerenciamento de permissões, com esses operadores podemos permitir ou negar certas operações em objetos de banco de dados.

Os operadores pertencentes ao grupo DCL são:

- *Grant* - concede ao usuário ou grupo de usuários permissões para realizar certas operações;
- *Revoke* - revoga as permissões concedidas;
- *Deny* - nega uma permissão a uma entidade de segurança.

4.6.3.4 Linguagem de Controle de Transação

Linguagem de Controle de Transação ou *Transact Control Language* (TCL) pertence a um grupo de operadores para gerenciar transações. Uma transação é um comando ou bloco de comandos que são completados com sucesso como um todo, com todas as alterações feitas na base de dados sendo corrigidas permanentemente ou canceladas, caso ocorra algum problema durante a transação, ou seja, todas as alterações feitas por qualquer comando incluído na transação serão canceladas.

Os operadores pertencentes ao grupo TCL são:

- *Begin Transaction* - determina o início de uma transação;
- *Commit Transaction* - aplica a transação;
- *Rollback Transaction* - reverter todas as alterações feitas no contexto da transação atual;

- *Save Transaction* - define o ponto de salvamento intermediário dentro da transação.

5 COMPUTAÇÃO EM NUVEM

5.1 Introdução

Para prosseguir com os próximos capítulos será necessário estabelecer alguns conhecimentos sobre o assunto de Computação em Nuvem, pois Banco de Dados como Serviço (tema do próximo capítulo) é, como o próprio nome diz, um serviço fornecido por meio de uma plataforma de serviços, ou seja, fornecido como serviço por um provedor na Nuvem.

Computação em Nuvem é um novo modelo arquitetural que está remodelando o setor de Tecnologia da Informação (TI). Acredita-se que nos próximos anos a TI será fornecida e entregue como serviço e isso é possível graças a virtualização dos recursos de *hardware*, possibilitando a utilização de tais recursos com mais eficiência. E isso acontece, pois a virtualização desacopla o ambiente de *software* do *hardware*. Com isso os servidores passam a existir como se fossem um único servidor em uma máquina virtual hospedada em um *DataCenter*.

Com a virtualização dos *DataCenters*, a Computação em Nuvem aumenta ainda mais o seu nível de eficiência. O fornecimento de infraestrutura, plataforma e serviço é entregue de forma mais fácil através de recursos de *pool*, conectividade universal e diversidade geográfica.

A Computação em Nuvem já se estabeleceu como uma nova forma de operar TI. E existem, além de Banco de Dados Como Serviço, diversas formas de uso que serão apresentadas neste capítulo.

5.2 Definição

Computação em Nuvem pode ser definido como um conjunto de recursos virtuais acessíveis que podem ser utilizados de forma fácil, tais recursos incluem *hardware*, *software*, plataformas, infraestrutura e serviços. E podem ser configurados dinamicamente, conforme as necessidades do momento, ou seja, varia de acordo com a carga de trabalho, permitindo a otimização do seu uso. Esse conjunto de recursos fornecidos são usualmente disponibilizados através de um modelo conhecido como

“Pague Pelo Uso”, ou seja, somente são pagos os recursos utilizados (VERAS, 2012, p.48)

Baseado na referência acima, pode-se dizer que a Computação em Nuvem tem como finalidade substituir ativos de TI por serviços. E conforme as necessidades computacionais forem aumentando, tais recursos serão disponibilizados mediante pagamento. Isso é possível, pois existem diversas tecnologias atuando em conjunto como a virtualização, arquiteturas de aplicação, infraestrutura orientadas a serviço, protocolos e tecnologias baseadas na Internet.

Segundo Veras, o objetivo da Computação em Nuvem é aprimorar a utilização de recursos de TI e torná-los mais econômicos, transferindo o risco da baixa utilização e da alta utilização para um cenário onde possa ser possível consumir recursos conforme a carga de trabalho. Onde as aplicações hospedadas nos *DataCenters* possam ser migradas para a Nuvem em um ambiente de larga escala e de recursos disponíveis de forma elástica. E para garantir a elasticidade requerida pelas organizações na utilização de recursos, é necessário que as aplicações hospedadas na Nuvem tenham uma arquitetura orientada a serviços, utilizando abordagens baseadas em técnicas de escalabilidade horizontal e escalabilidade vertical.

Atualmente a forma mais aceita de definir Computação em Nuvem é a estabelecida pelo *National Institute of Standards and Technology* (NIST) que se divide em cinco características essenciais, três modelos de serviços e quatro modelos de implantação (VERAS, 2012, p.50).

5.3 Características Essenciais

As quatro principais características que definem Computação em Nuvem são:

- Autoatendimento sob demanda: os recursos computacionais são providos de forma automática, sob demanda, sem a interação humana com o provedor de serviço.
- Amplo acesso a serviços de rede: os recursos computacionais são disponibilizados através da Internet e são acessados através de mecanismos padronizados, para que possam ser utilizados pelos dispositivos requerentes.

- *Pool* de recursos: os recursos computacionais do provedor são utilizados para servir a múltiplos usuários, sendo alocados e realocados dinamicamente conforme a necessidade.
- Elasticidade rápida: as funcionalidades computacionais devem ser rápidas e elasticamente providas, assim como rapidamente liberadas. O usuário dos recursos deve ter a impressão de que ele possui recursos ilimitados, que podem ser adquiridos (comprados) em qualquer quantidade e a qualquer momento. Elasticidade tem três principais componentes: escalabilidade linear, utilização *On-Demand* e pagamento por unidades consumidas de recursos.
- Serviços mensuráveis: os sistemas de gerenciamento utilizados pelos provedores de serviços controlam e monitoram automaticamente os recursos para cada tipo de serviço. Esse monitoramento do uso dos recursos deve ser transparente tanto para o fornecedor, quanto para o consumidor.

5.4 Modelos de Serviços

Os três principais modelos que definem Computação em Nuvem são IaaS, PaaS e SaaS que serão apresentados a seguir.

5.4.1 IaaS

Infraestrutura como Serviço, também conhecida como computação utilitária, é a capacidade que um provedor tem de oferecer uma infraestrutura física - que inclui CPU, memória, armazenamento, rede e energia elétrica - de forma transparente para o cliente. Esse serviço permite que os usuários tenham acesso, através de mecanismos de virtualização, a máquinas virtuais e com isso possam ter controle sobre o armazenamento, aplicações instaladas e até controle sobre os recursos computacionais de rede. Basicamente IaaS permite que o cliente deixe de adquirir *hardware* e *software* e foque somente em desenvolver a aplicação em uma infraestrutura virtualizada baseada na Internet e através do modelo “Pague Pelo Uso” (CUPPETT, 2016 ,p.126). *Google Cloud Platform, AWS* e *Azure* são alguns exemplos de provedores de IaaS.

5.4.2 PaaS

Plataforma como Serviço é o conjunto de ferramentas de desenvolvimento de *software* oferecidas por um provedor de serviços na Nuvem, onde os desenvolvedores podem usufruir de tais recursos por meio da rede de *Internet*. Essas ferramentas incluem uma variedade de sistemas operacionais, ambientes de desenvolvimentos de aplicações, contêineres e tecnologia de *middleware* (CUPPETT, 2016, p.126).

O modelo PaaS está relacionado com a utilização de uma plataforma de desenvolvimento de terceiros, onde rodam os aplicativos e se armazenam os dados. A grande diferença em relação a um modelo convencional de terceirização é que a plataforma roda em *DataCenters* de provedores externos.

Diferente do modelo IaaS que o usuário precisa estruturar e configurar máquinas virtuais, armazenamento e rede, o modelo PaaS já contempla todas essas questões. Além disso, Veras acrescenta que esse modelo possui as seguintes características (VERAS, 2012, p.200):

- Permite que o programador mantenha o foco no desenvolvimento das aplicações.
- Fornece um ambiente padrão independente das tecnologias utilizadas.
- Mantém o ambiente de *software* e sistema operacional.
- Propicia serviços prontos para uso que suportam as aplicações.
- Propicia um ambiente *On-Demand*.
- Pode ser construído para resistir a falhas.

5.4.3 SaaS

Software como Serviço é a entrega de serviços hospedados em uma plataforma na Nuvem que são suportados por uma infraestrutura (que inclui processamento, energia, e até um Banco de Dados, se necessário) e que são disponibilizados para os usuários interessados em tais recursos, isso considerando a compra de tais serviços através da plataforma. Essa é uma alternativa que aumenta a produtividade dos desenvolvedores, que não têm interesse em desenvolver, hospedar e configurar todo o ambiente de produção, e sim, somente, consumir o produto final. Isso faz com que SaaS seja a modalidade mais interessante dentre as ofertas “*as a Service*”, pois toda a responsabilidade de armazenamento, processamento e infraestrutura fica por conta

do provedor, dessa forma o usuário precisa somente se logar, com as suas credenciais de acesso, na plataforma e consumir os serviços, por meio de um dispositivo móvel, um navegador *web* ou uma aplicação *desktop* (CUPPETT, 2016 ,p.126).

Outra forma de consumir os serviços de um produto SaaS é por meio de *Application Programming Interface* (API). Caso o provedor disponibilize tal recurso, todas as informações e terminais para processamento ficam habilitados para que a aplicação do cliente possa se conectar e integrar com o seu fluxo de trabalho. E para garantir a segurança e integridade dos dados, as APIs, geralmente, possuem diversos mecanismos de segurança para autenticar e autorizar as aplicações dos seus clientes, assim como os seus usuários.

O SaaS pode ser considerado uma evolução do conceito de *Application Service Provider* (ASP), no sentido de oferecer serviços através da *Internet*, porém se forem comparados com a modalidade SaaS, o ASP tem mais similaridade com aplicativos hospedados de forma tradicional (*On-Premises*) do que produtos baseados no modelo SaaS. Isso por questões relacionadas a licenciamento, arquitetura e por inicialmente terem sidos projetados para atenderem um único cliente, ou seja, havia uma limitação na sua capacidade de compartilhar dados e processos com outras aplicações, somado a isso havia pouco benefício econômico em relação em relação ao SaaS (VERAS, 2012, p.223). *Google Apps* e o *SalesForce* são alguns exemplos de provedores de SaaS.

5.5 Variações “as a Service”

Apesar do conceito de Computação em Nuvem se basear em três práxis: Infraestrutura como Serviço, Plataforma como Serviço e Software como Serviço, existem diversos outros modelos da abordagem “*as a Service*”. Segundo Sharma, são em torno de sessenta e oito tipos, cada um com uma solução diferente para necessidades diferentes. O ecossistema de Computação em Nuvem tem um grande alcance e isso é compreendido pela comunidade científica que vem expandindo as vantagens dessa tecnologia para benefício de todos. A seguir serão listadas algumas das principais soluções “*as a Service*”.

- *DataBase as a Service*: transfere para o provedor boa parte da carga operacional de provisionamento, configuração, dimensionamento, ajuste de desempenho, *backup*, privacidade e controle de acesso, desde os usuários de Banco de Dados até a operadora de serviços, oferecendo custos totais mais baixos aos usuários. Os serviços fornecidos por um provedor de DBaaS garante a transferência da responsabilidade pelos dados e questões relacionadas ao gerenciamento, do cliente para o fornecedor. Além disso, o modelo DBaaS evita a necessidade de um Administrador de Banco de Dados que é o principal responsável pelo gerenciamento e manutenção de dados,minimizando os custos da organização. Com relação ao acesso, isso pode ser feito por meio de métodos muito simples, apenas chamadas de serviço e os usuários que utilizam esses serviços não sentem que estão interagindo com nenhum Banco de Dados externo. Mais detalhes serão abordados no próximo capítulo, mas para resumir, um DBaaS oferece: serviços de Banco de Dados em uma plataforma compartilhada e consolidada, autoatendimento para provisionamento de recursos, elasticidade para dimensionar e reduzir recursos de Banco de Dados e estorno com base no uso. Alguns exemplos: *Microsoft SQL Azure, Oracle Data Cloud, Cassandra* e *DynamoDB*.
- *Failure as a Service* (FaaS): permite que os serviços em nuvem executem rotineiramente simulações de falha em implantações reais. FaaS permite que os serviços em nuvem façam simulações de falha rotineiramente em larga escala, mantendo um processo organizacional e cultural de antecipar, mitigar e atuar em caso de falhas. Antes de ocorrer qualquer falha inesperada, um serviço em nuvem pode realizar simulações de vez em quando para descobrir nos cenários de implantação real, situações em que sua recuperação não funcionaria. Alguns exemplos: *Cloud Functions* e *Apache OpenWhisk.*
- *Storage as a Service* (StaaS): disponibiliza um espaço de armazenamento online na Nuvem para armazenamento de dados. Questões relacionadas à segurança e privacidade dos dados são solucionadas por criptografias. StaaS também recompensa os usuários pelo custo de computação otimizado, maior nível de segurança e sensibilidade com base na significância dos dados. Alguns exemplos: *OneDrive, Google Drive, DropBox* e *ICloud*.

- *Desktop as a Service* (DaaS): permite que os usuários gerenciem todo o seu trabalho por meio de um navegador seguro. Os usuários se conectam individualmente por meio de um navegador ao iniciarem seu trabalho. Esse serviço seguro facilita o acesso dos usuários aos arquivos, programas e *softwares*. Todos os recursos necessários para um funcionário trabalhar, em tese, estão contidos nessa plataforma. Um exemplo: *Citrix.*

5.6 Modelos de Implantação

Os quatros principais modelos de implantação de Computação em Nuvem são:

- Nuvem Privada: compreende uma infraestrutura de Computação em Nuvem desenvolvida e, na maioria dos casos, gerenciada pela organização. Os serviços são disponibilizados para serem consumidos pela própria organização, não estando publicamente disponíveis para uso geral. Em alguns casos pode ser gerenciada por terceiros. E basicamente existem dois tipos de nuvem privada: a hospedada pela empresa e a hospedada em um provedor de serviço
- Nuvem Pública: é disponibilizada publicamente através do modelo “Pague Pelo Uso”. São desenvolvidas e disponibilizadas por organizações públicas.
- Nuvem Comunitária: nesse modelo, a infraestrutura de Computação em Nuvem é compartilhada por diversas organizações e suportada por uma comunidade que possui interesses comuns. A nuvem comunitária pode ser gerenciada pelas organizações que fazem parte da comunidade ou por terceiros.
- Nuvem Híbrida: a infraestrutura é uma combinação de dois ou mais modelos de nuvens, conectadas através de um mecanismo próprio ou padronizado que favorece a portabilidade de dados e aplicações.

6 BANCO DE DADOS COMO SERVIÇO

6.1 Introdução

Como vimos no capítulo anterior, recursos da *Web* como infraestrutura, plataforma e *software* passaram a serem disponibilizados por empresas terceiras, facilitando boa parte do trabalho do desenvolvedor de *software*, caso opte pelo uso desses recursos. Essa mesma tendência se estendeu para Banco de Dados, movendo todo o seu gerenciamento para uma plataforma na Nuvem, uma arquitetura terceirizada. Dessa forma o proprietário dos dados migra ou implanta todos os seus dados para um provedor terceirizado gerenciar.

Neste capítulo será apresentado com mais detalhes o conceito e os aspectos que constituem DBaaS, uma recente e importante implementação "*as a Service*" que tem como proposta a transformação dos mecanismos de gerenciamento de Banco de Dados, trazendo mais flexibilidade para o ambiente organizacional, com recursos automatizados como de cópia de segurança, escalonamento, gerenciamento de ciclo de vida entre outros. O que torna mais simples a forma de gerenciar os dados e, em alguns casos, diminuindo custos desnecessários para a empresa.

6.2 Definição

Banco de Dados como Serviço, também conhecido como DBaaS do termo *Database as a Service*, é uma oferta de serviço gerenciado de computação em Nuvem que fornece acesso aos usuários a um Banco de Dados sem a necessidade de configuração de *hardware* físico, a instalação de *software* ou a necessidade de configurar o Banco de Dados. Todas as tarefas administrativas e de manutenção são realizadas pelo provedor de serviços, dispensando os usuários dessas tarefas para usufruírem do uso do BD (CUPPETT, 2012, p.129).

DBaaS é um dos modelos de serviço secundários da computação em Nuvem. Podendo ser considerado uma subespecialidade do modelo SaaS. Em suma, DBaaS é um serviço gerenciado que oferece acesso a um Banco de Dados para ser usado com aplicativos e seus dados relacionados. Esse aplicativo é acessível ao usuário, geralmente, por meio de um aplicativo da *web*, que o usuário pode usar para gerenciar

e configurar o Banco de Dados e até mesmo provisionar ou desprovisionar instâncias de Banco de Dados. Neste modelo, o pagamento pode ser cobrado de acordo com a capacidade utilizada, bem como as funcionalidades e utilização das ferramentas de administração de Banco de Dados.

6.3 Arquitetura

Com o crescimento do volume de dados que os SGBD passaram a processar, surgiu a necessidade de uma mudança arquitetural na forma como os sistemas relacionais processavam os dados. Esse grande volume resultou em problemas como inconsistência e indisponibilidade nos BD distribuídos, que tem como finalidade minimizar tais problemas, pois quanto mais distribuído é um BD maior será a escala de relacionamento, reduzindo o desempenho do sistema à medida que o número de partições aumentam (LOWE, 2021, p.6).

Esse problema resultou no retorno do conceito do *NoSQL*, ideia proposta na década de 1960, que dispensa o modelo relacional e se baseia no modelo chave/valor, um modelo básico de organização de armazenamento de dados. A tabela em um BD não relacional é composta por duas colunas, uma chave e um valor, onde o resultado das consultas é um valor não definido. Esse fato fez com que tal conceito fosse dispensado no seu surgimento e substituído pelo modelo relacional (KRIEGEL, 2011, p.331).

Pelo fato do modelo não relacional do padrão *NoSQL* não necessitar de diversas tabelas para armazenamento de dados, há um barateamento dos custos totais para esse tipo de arquitetura, tornando esse padrão bem difundido no modelo de armazenamento em Nuvem.

6.4 Arquitetura de armazenamento

O *design* arquitetural de armazenamento para DBaaS possui algumas características essenciais como escalabilidade e segurança. A seguir serão apresentados alguns modelos de arquitetura de armazenamento que podem ser adotados para um DBaaS.

6.4.1 Arquitetura em camadas

Com o crescimento na procura pela terceirização do gerenciamento de dados, surgiu a necessidade do desenvolvimento de uma arquitetura específica para atender o modelo de armazenamento em Nuvem. Para atender tais necessidades, em 2012 foi proposto o modelo de arquitetura em camadas específico para SGBD baseado em Nuvem. A arquitetura em camadas facilita adicionar mais funcionalidades na aplicação, facilita no processo de manutenção e distribui a ameaça à segurança, diminuindo os potenciais risco de invasão (MEHAK; MASOOD; GHAZI; SHIBLI; KHAN, 2014, p.8).

Essa arquitetura se divide em quatro camadas, que são:

- Camada de Interface de Usuário: utilizada pelos usuários para terem acesso aos serviços.
- Camadas de Aplicação: utilizada para acessar os serviços de *software* e espaço de armazenamento em Nuvem.
- Camada de Banco de Dados: fornece mecanismos de gerenciamento de Banco de Dados, utilizando instruções de consulta e processamento.
- Camada de Armazenamento: criptografa os dados armazenados, realiza processos de *backup* e monitoramento de disco.

6.4.2 Shared-Disk

Shared-Disk é uma arquitetura de computação distribuída que utiliza a sua própria memória em conjunto com outros dispositivos conectados, onde cada “nó” tem a sua própria memória e também permite o compartilhamento. Dessa forma, caso ocorra alguma falha em um “nó” na rede compartilhada, o dispositivo pode ter acesso a memória de outros dispositivos conectados. Diferentes processos de Banco de Dados têm acesso a todos os recursos da rede compartilhada, assim qualquer servidor pode fornecer o serviço de armazenamento, caso necessário, mediante solicitação. As vantagens dessa arquitetura são que cada processador tem sua própria memória, o barramento de memória não se torna um empecilho e o sistema oferece uma maneira simples de fornecer um certo grau de tolerância a falhas (MEHAK; MASOOD; GHAZI; SHIBLI; KHAN, 2014, p.9).

6.4.3 Shared-Nothing

Shared-Nothing é uma arquitetura de computação distribuída que cada dispositivo possui a sua própria memória em um ou mais discos locais, cada solicitação de atualização é atendida por um único nó em um *cluster* de computador e a comunicação dos processadores em *cluster* é feita por meio de mensagens na rede.Todas as solicitações são roteadas automaticamente para o sistema e apenas um *cluster* por vez pode possuir e consumir recursos. Assim, o problema de consistência de dados é evitado. Os “nós” não compartilham a mesma memória, assim os pontos de falhas são eliminados, permitindo que o sistema geral continue operando apesar das falhas em nós individuais e permitindo que nós individuais sejam atualizados sem um desligamento de todo o sistema (MEHAK; MASOOD; GHAZI; SHIBLI; KHAN, 2014, p.9).

6.5 Segurança

Um dos aspectos mais importantes sobre DBaaS é a questão da segurança. Manter os dados protegidos de invasores é algo fundamental para uma organização. Os riscos que envolvem a segurança dos dados incluem o gerenciamento de dados, confidencialidade, integridade, disponibilidade entre outros. Como os dados passam a ser responsabilidade do provedor do serviço de Banco de Dados, questões como a integridade, confidencialidade e disponibilidade passam a ser comprometidas, sendo necessário que o provedor do serviço evidencie quais são os padrões de segurança estabelecidos (MEHAK; MASOOD; GHAZI; SHIBLI; KHAN, 2014, p.12). A seguir serão apresentados, baseado na referência citada, os principais pontos referentes à segurança em DBaaS e quais são os melhores mecanismos para evitar tais riscos.

6.5.1 Confidencialidade

No contexto de DBaaS, confidencialidade se refere a segurança das consultas que serão executadas em uma plataforma terceirizada. Diversos problemas relacionados à segurança podem ocorrer se os dados não forem criptografados no Banco de Dados.

Assim que os dados forem migrados para a nuvem, o proprietário dos dados não consegue ter acesso a política de controle de acesso. Qualquer usuário administrador do provedor de serviço pode ter acessos a todos os seus dados. Isso ocorre, pois geralmente quem controla os acessos é o provedor do serviço e não o proprietário dos dados.

Os usuários de Banco de Dados que possuem um perfil de administrador também podem ser um risco à segurança dos dados, caso o mesmo utilize tal privilégio de forma maliciosa ou irresponsável. E vazamentos de dados sigilosos podem acontecer nesse caso. Para evitar e/ou minimizar esse problema interno, o provedor de serviço pode submeter, no momento da contratação, o futuro superusuário a entrevistas mais específicas, assim como especificar em contrato quais serão as consequências legais se alguma das cláusulas forem infringidas.

Além dos possíveis riscos internos, existem também os riscos externos. Ataques como *phishing*, *scamming* e exploração de vulnerabilidade são problemas de segurança enfrentados em DBaaS. Um usuário malicioso pode explorar alguma vulnerabilidade na segurança do provedor de serviço através de ataques *spoofing*, *sniffing*, *man-in-the-middle*, entre outros. Assim os dados dos clientes podem ficar sujeitos a esse tipo de vulnerabilidade.

Provedores de DbaaS geralmente mantêm as cópias de segurança dos *backups* periódicos em locais físicos distintos, a fim de manterem a replicação dos dados, e caso ocorra algum problema no dispositivo de armazenamento principal, algum dos dispositivos secundários entram para substituí-los. Também há o fornecimento de serviços terceirizados aos provedores, em casos mais específicos. Assim, tanto as réplicas de segurança, quanto aos prestadores terceirizados, devem ter a mesma atenção na questão da confidencialidade dos dados.

Para reduzir os problemas relacionados à confidencialidade dos dados, um algoritmo de distribuição e compartilhamento redundante pode ser a base de solução para esse problema. Esse algoritmo divide e distribui entre servidores diferentes cada valor de atributo. Nesse cenário há dois servidores. De forma resumida, um serve para manter os dados pesquisados e outro para manter os índices incrementais da consulta. Além disso, todos os valores são criptografados no servidor.

6.5.2 Privacidade

A privacidade da informação é um assunto que está diretamente relacionado com DBaaS, pois a partir do momento que se terceiriza a responsabilidade dos dados de uma organização, assim como os de seus clientes, há uma perda no controle da divulgação desses dados, no sentido de que tais dados podem ser divulgados e/ou vendidos, para benefício da empresa provedora do armazenamento.

Além disso, há o problema de que as informações terceirizadas podem estar armazenadas em diversos servidores, e distribuídas em diversos locais físicos. Com isso, cada servidor de dados vai estar sujeito às estruturas jurídicas do país em questão, estando sujeito a diversas restrições, dependendo de cada país.

Dessa forma, cabe ao contratante do serviço saber avaliar bem quais são os riscos de se manter os seus dados e os de seus clientes em uma plataforma na Nuvem. E para minimizar os problemas de privacidade, também pode-se utilizar algoritmos de criptografia. Um serviço de criptografia que pode ser utilizado é o *NetDB2*. Esse serviço é fornecido via API e os usuários podem utilizá-los por meio de um navegador *web*. Ele se baseia nos protocolos de segurança *Transport Layer Security* (TLS) e *Secure Sockets Layer* (SSL). Seu princípio de funcionamento consiste em compactar densamente os valores dos dados em um bloco de criptografia e realizar o processamento diretamente no texto cifrado usando um esquema de criptografia homomórfico seguro. A segurança é garantida porque o servidor de Banco de Dados executa a maior parte da computação sem ter acesso à chave secreta ou aos dados confidenciais (MEHAK; MASOOD; GHAZI; SHIBLI; KHAN, 2014, p.20).

6.5.3 Integridade

A integridade em DBaaS se refere à proteção dos dados contra modificação, exclusão ou inserção indevida. Esse risco pode ocorrer quando por meio de um usuário administrador mal intencionado. Por ter total acesso a base de dados e permissão para executar determinados comandos, ele pode manipular os dados, sem nenhuma restrição. Assim como se vier a ocorrer uma invasão hacker, os dados também podem estar em risco.

Além disso, os provedores de serviços em Nuvem fornecem aos seus clientes arquivos de configuração de acesso. Esses arquivos contêm especificações de níveis de acesso. Se ocorrer algum problema nesses arquivos, consequentemente isso impactará no pleno funcionamento do Banco de Dados e na forma como os dados serão entregues (MEHAK; MASOOD; GHAZI; SHIBLI; KHAN, 2014, p.20).

Em vista dessa vulnerabilidade, foi proposto um mecanismo de auditoria de integridade que se baseia em dados criptografados por eliminação distribuída e encriptação homomórfica. Essa técnica permite que os auditores e os usuários auditem as transações que ocorreram no Banco de Dados em um determinado período, através de um arquivo de log, por meio de um protocolo de comunicação de baixo nível de processamento. Com o resultado da auditoria é possível saber quais foram as transações que ocorreram no Banco de Dados e por quais usuários essas transações foram executadas, assim como quais erros vieram a ocorrer e qual foi o motivo. Pelo fato dos auditores não terem acesso a chave privada, não há a possibilidade de saberem o conteúdo das informações, garantindo a integridade dos dados do cliente (MEHAK; MASOOD; GHAZI; SHIBLI; KHAN, 2014, p.23).

6.5.4 Disponibilidade

A disponibilidade em DBaaS se refere o quão acessíveis os dados são para os seus clientes. Esse é um dos principais requisitos de segurança se tratando de computação em Nuvem. Eventos que podem fazer um DBaaS ficar indisponível são desastres naturais, falhas de equipamentos, invasões maliciosas, entre outros. E a indisponibilidade pode ser temporária ou permanente, dependendo da situação e dos mecanismos de atuação em casos de falha, porém independente da solução, qualquer indisponibilidade irá afetar o cliente em algum nível, principalmente se a falha ocorrer em horário comercial.

Problemas relacionados com o tráfego de dados podem afetar diretamente os serviços de um provedor de DBaas. Transações envolvendo muitos pacotes podem resultar em uma alta latência, fazendo com que o usuário não tenha uma experiência agradável. Isso sem considerar o tempo de processamento de uma consulta ou processo no Banco de Dados (CUPPETT, 2016, p.131). Configurações incorretas de rede e falta de isolamento de recursos são exemplos que podem agravar a latência

de dados. Para alguns casos existem protocolos que tratam a questão da latência, porém isso depende da análise técnica do profissional que irá implementar a rede.

O consumo total dos recursos pode levar a indisponibilidade do serviço de DBaaS. Se um cliente utilizar todos os recursos disponíveis, não haverá mais recursos necessários para continuar os acessos aos serviços. Pelo fato de não ter mais "crédito", um usuário pode, após ter esgotado seus recursos, tentar acessar o serviço de DBaaS e ter sua solicitação de requisição negada. Pelo fato dos recursos serem limitados a uma quantidade prevista em contrato, o cliente deve se precaver no momento em que consome os dados do provedor. Se for necessário, um novo contrato pode ser estabelecido, aumentando seu limite.

O sincronismo de *backups* é um problema de gerenciamento de inconsistência que pode afetar a disponibilidade dos dados. Dependendo do tamanho do Banco de Dados, do tamanho das requisições feitas e da quantidade de usuários acessando os dados, a resposta pode levar algum tempo até ser processada. Os provedores de DBaaS tentam equilibrar os princípios de consistência, disponibilidade e partição, porém nunca esses fatores são atendidos com a máxima eficiência. Dessa forma, para manter os *backups* periódicos e a consistência no resultado das consultas, a performance acaba sendo comprometida.

Bloqueio de dados pode se tornar um problema de disponibilidade, caso o provedor do serviço de DBaaS dificulte a migração dos dados de seus clientes para outra plataforma. Essa abordagem é conhecida como "*Cloud Lock-In*", e basicamente isso acontece quando um cliente deseja migrar seus dados para um provedor concorrente, onde o provedor aumenta as taxas cobradas para realizar a migração. A AWS é uma empresa que é conhecida por realizar essa prática. Esse tipo de problema pode ser evitado no momento da contratação do serviço, analisando se há vantagem correr esse risco.

Outro problema semelhante ao citado anteriormente é o "*Data Lock-In*". Esse problema ocorre quando um provedor de DBaaS tem os dados de seus clientes migrados para outro provedor, porém sem que haja alguma padronização para armazenamento de dados, ocasionando o bloqueio de dados. Um exemplo disso, foi o caso que ocorreu com as empresas *Linkup* e *Nirvanix*. A *Linkup* encerrou seus serviços em 2008 e passou a responsabilidade do armazenamento dos dados dos seus clientes para a *Nirvanix*, porém como não havia uma padronização de armazenamento, os clientes ficaram impossibilitados de acessar o serviço. Uma

possível solução para esse problema de bloqueio de dados, seria uma API para padronizar o armazenamento, processamento e extração de dados em DBaaS (MEHAK; MASOOD; GHAZI; SHIBLI; KHAN, 2014, p.16).

Desastres naturais são também um risco que pode ocasionar na indisponibilidade do serviço. Esse risco pode resultar em graves consequências para o desempenho ou até comprometer totalmente o fornecimento do serviço. Tempestades, raios e terremotos são alguns exemplos de desastres naturais que podem prejudicar o provedor de DBaaS.

Para garantir a acessibilidade dos dados, diversos servidores para armazenamento devem estar a disposição para realizar os processos periódicos de *backup*. Caso ocorra alguma falha em algum servidor, imediatamente entrará outro para substituí-lo, sem comprometer o processamento ou extração dos dados. Os dados armazenados nesses servidores, tanto o principal quanto o de *backup*, devem ser criptografados por meio de uma chave secreta e somente descriptografados após a extração (MEHAK; MASOOD; GHAZI; SHIBLI; KHAN, 2014, p.24).

6.6 Configuração

Os provedores de DBaas possuem uma característica em comum, que é a limitação de configuração. Diferente de um BD *On-Premises*, onde o administrador de BD pode ter acesso total às configurações de desempenho do banco, em um DBaaS essa possibilidade fica limitada às opções que o provedor vier a oferecer. Configurações de rede, taxa de transferência de armazenamento e *kernel* são alguns exemplos de configurações que um Administrador de Banco de Dados poderia ajustar, caso o provedor fornecesse, que aumentaria o desempenho do serviço. Mesmo que as configurações básicas disponíveis para o administrador simplifique o trabalho de configuração, o ajuste fino seria a opção mais viável, principalmente em cenários envolvendo uma aplicação muito robusta, com um volume grande de dados para processar e com inúmeras requisições ao servidor.

Por outro lado os DBaaS possuem uma característica vantajosa que é a de escalabilidade automática. Essa solução permite que seja adicionado mais recursos de processamento e memória em tempo real, conforme as necessidades dos clientes. E isso é possível fazer através de configurações, em um painel de otimização de

recursos, sem a necessidade de intervenção humana. Por exemplo, a memória pode ser aumentada em N% se a memória livre cair abaixo 5%. Ou 0,25% de CPU pode ser adicionado quando o uso da CPU excede 98% por mais de 5 minutos (CUPPETT, 2016, p.132). Porém algo a se atentar é que os recursos disponíveis de CPU e armazenamento são limitados a uma quantidade máxima estabelecida em contrato, caso o cliente exceda o seu limite de recursos, ele será impedido de consumir os serviços.

6.7 Monitoramento

O monitoramento de transações é uma questão de suma importância quando se fala em DBaaS. Saber quais processos foram realizados com sucesso ou com falha determina o quão bem projetado foi o Banco de Dados, assim como a qualidade do serviço fornecido pelo provedor. Ter a informação de qual momento e qual o motivo que ocasionou uma falha no Banco de Dados, em um processamento ou extração, pode ser uma informação crucial para que o Administrador de Banco de Dados atue na causa raiz e consiga solucionar o problema da forma mais rápida e eficaz possível, sem consequências para o pleno funcionamento da aplicação. Por esse motivo, ao se contratar um serviço de armazenamento na nuvem, é recomendado que o cliente saiba se o provedor possui algum mecanismo informando quais foram as falhas ocorridas no Banco de Dados, para que o mesmo atue da melhor forma possível, pois mesmo que o provedor do serviço seja responsável pelo armazenamento dos dados, o administrador também tem responsabilidade na supervisão e atuação, caso seja necessário.

Além de estar a parte de como estão os processos no Banco de Dados, o administrador deve-se atentar se o provedor mantém muitos outros clientes no seu *DataCenter*, pois isso pode ser um fator que pode comprometer a qualidade do serviço fornecido. Hospedar diversos clientes em um mesmo *DataCenter* pode sobrecarregar a capacidade de processamento do servidor e comprometer não somente o cliente que fez a requisição, mas também os demais clientes hospedados naquele mesmo *DataCenter*. Se um cliente possui muitos dados em seu Banco de Dados e realiza com muita frequência requisições para processamento e extração de dados, isso pode prejudicar o pleno funcionamento dos demais aplicativos consumidores daquele BD.

Recursos como energia, rede e processamento são compartilhados entre vários clientes, caso o provedor seja de co-localização. Esse problema existente em *DataCenters* é minimizado através de uma separação modular entre os DBaaS (CUPPETT, 2016, p.134). Com essa abordagem, a sobrecarga no servidor causada por um determinado cliente, não afetará outro cliente.

6.8 Integração Contínua

A Integração Contínua em DBaaS é semelhante a realizada em um Banco de Dados *On-Premises*. As alterações feitas na aplicação devem ser replicadas para o Banco de Dados, caso necessário, ou de forma manual ou automática. Assim a integração entre ambas as partes pode ser realizada de forma segura, sem que haja conflito de versionamento.

6.9 Vantagens

Uma solução DBaaS permite que a organização transfira todo seu Banco de Dados para uma empresa terceira gerenciar, de forma ágil e flexível, simplificando as tarefas que até então eram responsabilidades do Administrador de Banco de Dados.

Os custos relacionados à implementação de um Banco de Dados na Nuvem, em alguns casos, são bem menores do que uma solução *On-Premises*. E não há a necessidade de planejamento com antecedência e de elaborar cálculos de custos referentes a aquisição de *hardware* e *software* junto com o departamento de compras. Os provedores de DBaaS conseguem estimar qual seria o contrato ideal para cada situação, dependendo das necessidades do cliente. Caso seja para atender uma aplicação mais simples, com poucas tabelas no BD, um contrato disponibilizando menos recursos computacionais já poderá atender. Se for para atender uma aplicação mais robusta, com muitas tabelas no BD e com muitas requisições de extração e processamento para o servidor, então um contrato disponibilizando mais recursos seria o mais ideal.

Todos os processos pertinentes a BD são feitos de forma automática, por meio de agendamento e configuração, e é possível ter uma visão geral de tudo que aconteceu e está acontecendo no BD. Provisionamento de instâncias, criação e

agendamento de *backup* e alerta de falhas são exemplos de tarefas que seriam manuais, e de certa forma até trabalhosas, que através dos mecanismos que o provedor disponibiliza podem ser realizadas e analisadas de forma mais simples.

Os *backups* periódicos permitem que caso ocorra alguma falha no dispositivo de armazenamento principal, entre outro imediatamente para substituí-lo sem intervenção humana e sem grandes consequências para o cliente, mantendo sempre a alta disponibilidade do serviço. Também pode ser configurado o *backup* de dados sensíveis, caso algum dado seja alterado indevidamente por algum usuário, esse dado pode ser recuperado.

A criptografia dos dados armazenados na Nuvem garante que as informações referentes aos seus clientes não serão divulgadas e nem comercializadas para interesse de terceiros. Uma vez que os dados estão criptografados no Banco de Dados, somente o cliente consegue ter acesso a informação real que está armazenada. Isso é possível pois somente o cliente tem acesso a chave privada que possibilita a descriptografia dos dados no momento da extração ou do processamento, assim é mantida a privacidade do cliente.

Caso seja necessário mais recurso de processamento e memória, o próprio provedor disponibiliza para o cliente, de forma automatizada e sem a intervenção humana. Nesse cenário não há a necessidade de adquirir mais recursos computacionais para atender uma necessidade, que em alguns casos podem ser pontuais. Um exemplo disso é o caso onde um comércio eletrônico deseja realizar um evento promocional por um curto período de tempo. É esperado que nesse intervalo de tempo haja uma sobrecarga de trabalho no servidor, devido aos inúmeros acessos. Para atender essa situação de forma simples, sem que haja a necessidade de comprar, instalar e configurar recursos de processamento e memória computacional, basta contratar mais recursos para atender essa situação e posterior a isso voltar para o contrato mais em conta.

E questões relacionadas a manutenção de equipamentos, em uma solução DBaaS, são completamente transparentes para o cliente. Não é necessário manter ou contratar uma equipe para realizar os processos de manutenção de *hardware* e nem atualizar o *software* de gerenciamento de Banco de Dados e sistema operacional. Toda essa responsabilidade fica por conta do provedor e o cliente fica livre para poder desenvolver a aplicação e discutir regras de negócio.

6.10 Desvantagens

DBaaS possui algumas desvantagens que algumas organizações podem considerar como sendo um problema na hora de escolher onde será hospedado o seu Banco de Dados.

A questão da segurança e privacidade podem ser pontos questionáveis em um provedor de DBaaS. Mesmo que o fornecedor do serviço garanta para o cliente que os seus dados estão armazenados de forma segura, não há como ter a certeza disso, pois o cliente tem acesso somente às configurações básicas disponíveis na plataforma, como um usuário comum. Se algum funcionário interno, com acesso ilimitado de recursos, tiver a intenção de prejudicar algum cliente, possivelmente ele conseguirá. E em cenário de invasão *hacker* no *DataCenter*, não há nada que o cliente possa fazer, isso se de alguma forma ele ficar sabendo. Nesse caso o cliente fica somente confiando nos requisitos de segurança que o provedor diz cumprir.

O controle sobre as configurações mais específicas relacionadas ao desempenho do Banco de Dados também ficam restritos ao provedor de DBaaS. Caso o administrador receba uma reclamação dos usuários alegando que a aplicação está com lentidão por conta do Banco de Dados, não há como ter acesso às informações mais precisas sobre o que está acontecendo. Saber quais consultas e quais processos estão onerando o servidor de armazenamento ficam restritos somente para o provedor, o administrador não saberia informar se o problema da lentidão está sendo causada no servidor da aplicação ou no provedor de DBaaS. Encerrar possíveis consultas e processos lentos também ficam fora da alçada do administrador.

A banda larga e a configuração de rede são questões que impactam diretamente no desempenho de uma aplicação integrada com um DBaaS. O tráfego de dados deve ser bem avaliado antes de se contratar um serviço na Nuvem, ainda mais se tratando de Banco de Dados. A saturação no uso da rede de dados influencia na latência e isso acaba refletindo na satisfação do usuário com o sistema, mesmo isso não sendo uma responsabilidade nem do desenvolvedor nem do provedor. Por esse motivo, em alguns casos, pode não ser vantajoso migrar um BD para a Nuvem, devido ao grande consumo de banda larga.

Outro problema que os clientes que possuem seus dados na Nuvem podem enfrentar é o bloqueio de dados. Como citado anteriormente, existem basicamente duas formas do cliente ter seus dados bloqueados por um provedor na Nuvem, o

Cloud Lock-In e o *Data Lock-In*, e nesses dois casos não há muito o que o cliente fazer, ou ele paga as taxas pertinentes a migração de dados para outro provedor ou, no caso do Data Lock-in, aguarda até que o provedor desenvolva uma solução para atender às suas necessidades. Caso isso não ocorra, o cliente simplesmente fica impossibilitado de ter acesso aos seus dados.

7 Estudo de caso: Criação de Banco de Dados como Serviço na plataforma Heroku

7.1 Introdução

Para demonstrar a criação e utilização de um DBaaS foi escolhida a plataforma de serviços na nuvem *Heroku* e o SGBD *PostgreSQL*. Por meio dessa plataforma é possível criar e consumir um Banco de Dados na Nuvem, sem a necessidade de adquirir um servidor ou uma máquina virtual, todas essas questões ficam por responsabilidade do provedor. A demonstração será feita por meio da interface gráfica *DBeaver* e consistirá na criação de um Banco de Dados por meio de comandos DDL e DML, para criar as estruturas de armazenamento, manipulação e extração de dados.

7.2 Heroku

Heroku é uma plataforma de serviços na Nuvem (PaaS) que permite aos desenvolvedores realizar implantação, escalonamento e gerenciamento de aplicativos. Esta plataforma oferece suporte para uma ampla gama de linguagens de programação, como *Java, Ruby, PHP, Node.js, Python, Scala* e *Clojure*. O *Heroku* executa aplicativos por meio de contêineres virtuais conhecidos como *Dynos*.E a cobrança é feita com base no número de máquinas virtuais que são necessárias para os aplicativos. A plataforma *Heroku* e os aplicativos criados pelo usuário usam a AWS como infraestrutura subjacente.

7.3 Heroku Postgres

Heroku Postgres é uma oferta SGBD. Os Bancos de Dados *PostgreSQL* podem ser acessados por meio de um *driver PostgreSQL* em todas as linguagens suportadas pelo *Heroku*.

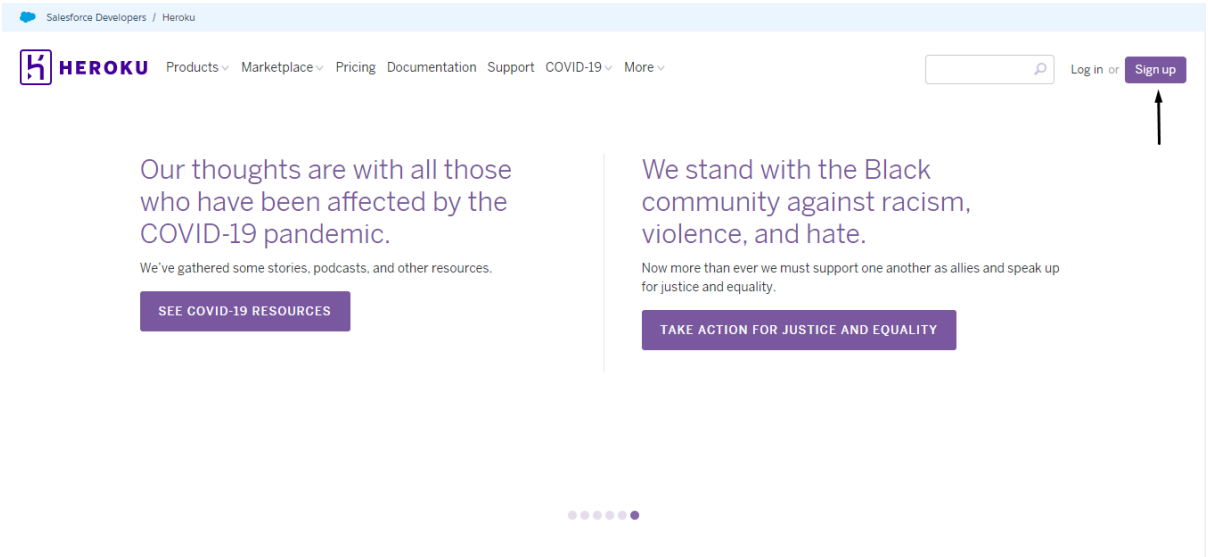
7.4 DBeaver

DBeaver é um *software* cliente de Sistema de Gerenciamento de Banco de Dados Relacionais (SGBDR) e usa a API do JDBC para interagir com Bancos de Dados por meio de um *driver* JDBC. Para outros Bancos de Dados, por exemplo o NoSQL, o *DBeaver* usa *drivers* de Banco de Dados proprietários. E fornece um editor que suporta o preenchimento de código e o realce de sintaxe. Este *software* foi escrito em Java e baseado na *Integrated Development Environment* (IDE) Eclipse.

7.5 Login na plataforma Heroku

Para acessar a plataforma Heroku é necessário entrar no site "https://www.heroku.com" e clicar no botão "*Sign up*", evidenciado na Figura 1.

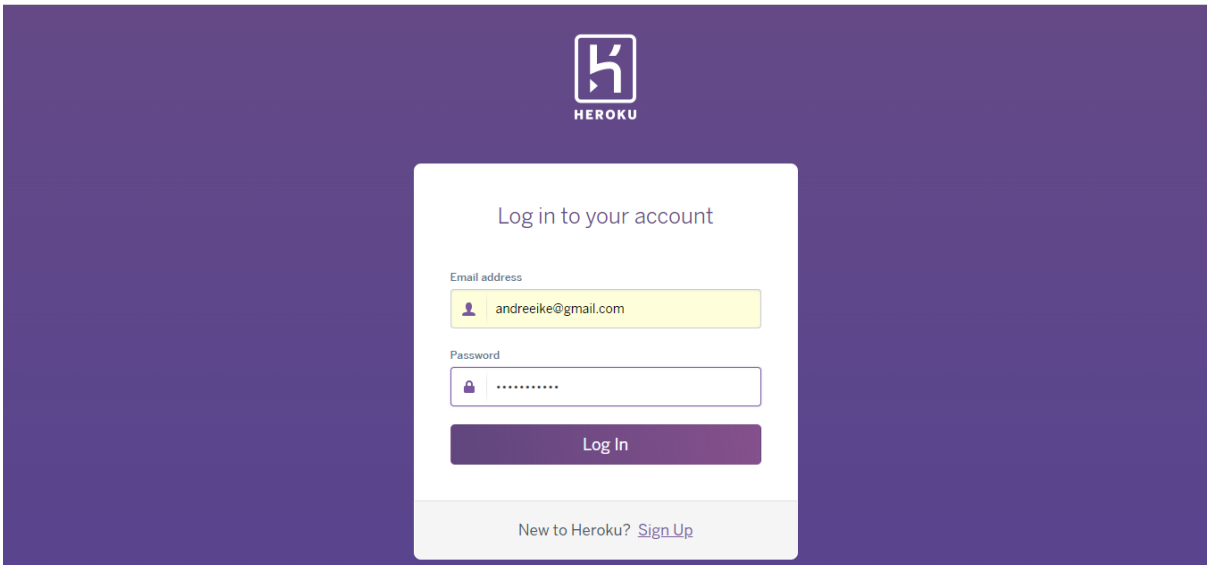
Figura 1 - Página inicial.



Fonte: print screen da plataforma Heroku.

Após isso o usuário será redirecionamento para a seguinte página evidenciada na Figura 2.

Figura 2 - Página de login.

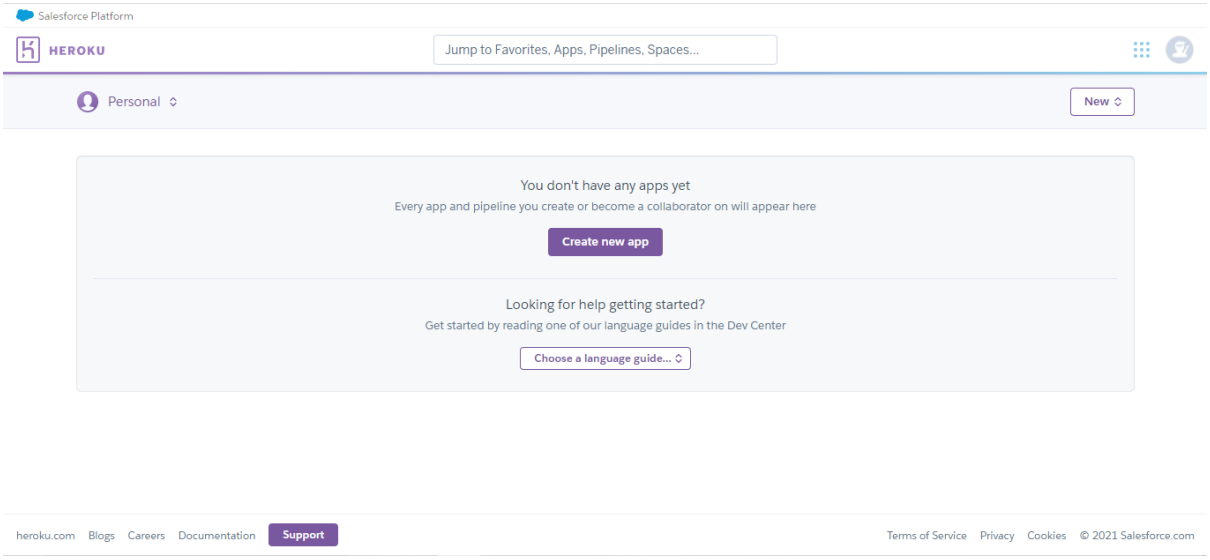


Fonte: print screen da plataforma Heroku.

O usuário deverá inserir suas credenciais de acesso e clicar no botão “*Log in*”. Caso não possua acesso, deverá clicar no link “*New to Heroku?*” para criar um novo acesso.

Feito o *login*, o usuário será direcionado para página de aplicação, conforme Figura 3.

Figura 3 - Página para criar novo aplicativo.

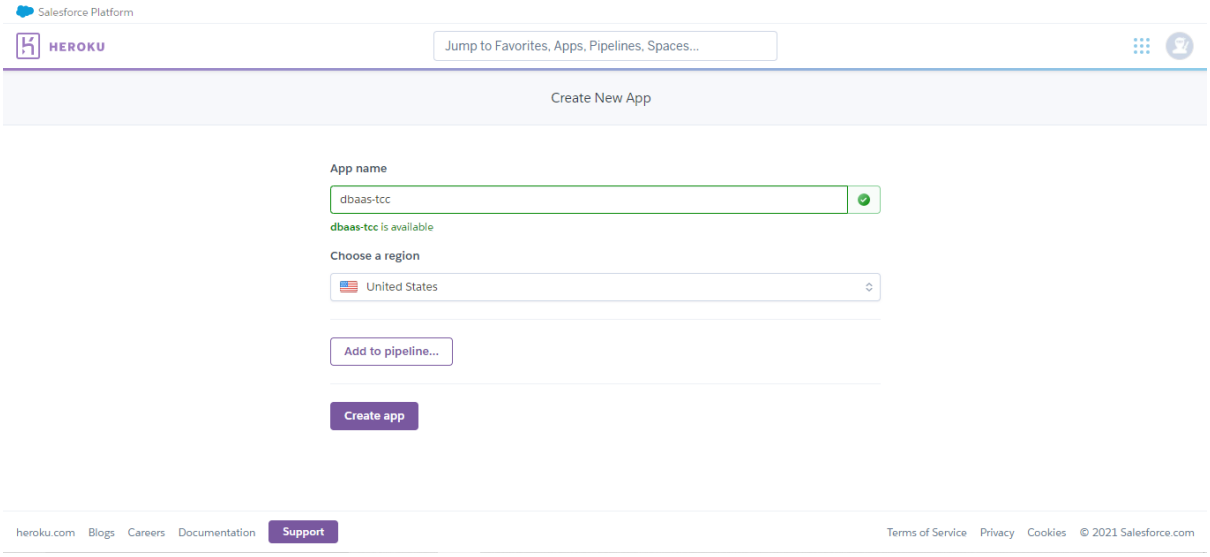


Fonte: print screen da plataforma Heroku.

7.6 Criação da aplicação

Para criar uma aplicação, clicar no botão “*Create new app*”, conforme a Figura 3. Assim o usuário será direcionado para a seguinte página evidenciada na Figura 4.

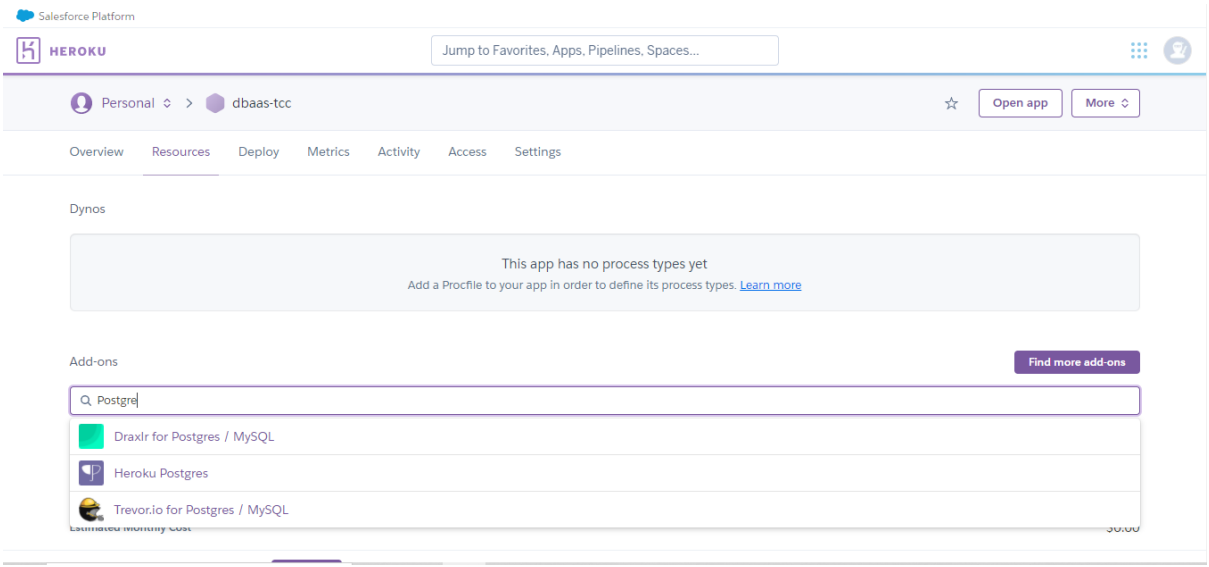
Figura 4 - Criação da aplicação.



Fonte: print screen da plataforma Heroku.

Será solicitado o nome da aplicação e a região. O nome escolhido foi “dbaas-tcc” e a região foi “*United States*”. Após isso, clicar no botão “*Create app*”. Em seguida o usuário será direcionado para a página evidenciada na Figura 5.

Figura 5 - Localizando Bancos de Dados *PostgreSQL.*



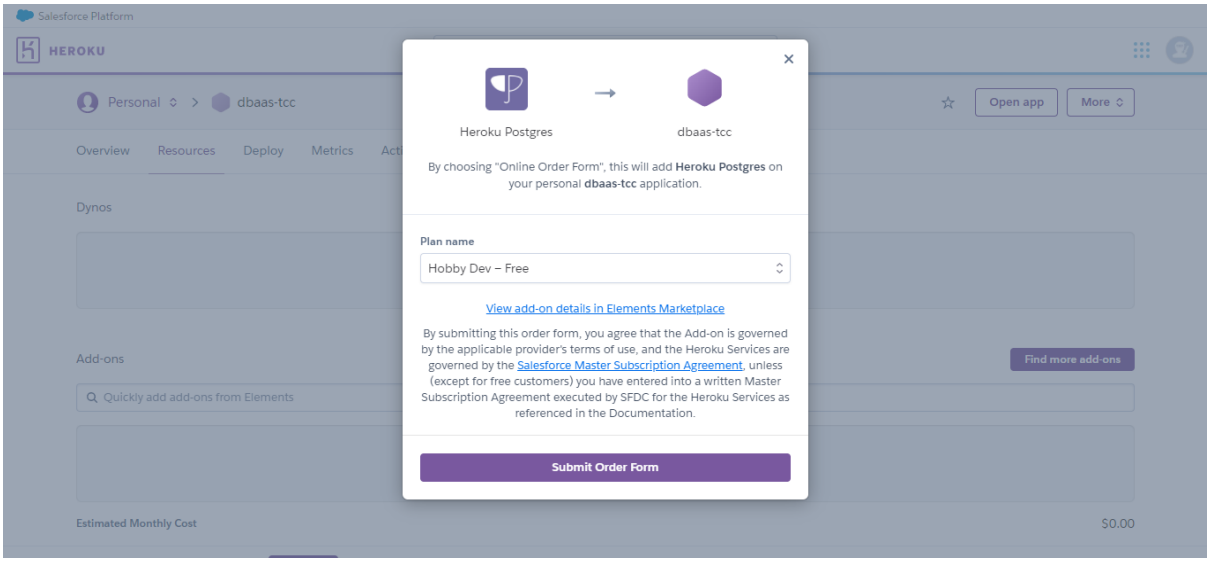
Fonte: print screen da plataforma Heroku.

7.7 Criação do DBaaS

Para adicionar o DBaaS na aplicação, é necessário clicar na aba “*Resources*” e no campo “*Add-ons*”, pesquisar por “*Heroku Postgres*”, conforme a Figura 5.

Informar o plano para o serviço de DBaaS. Foi escolhido o plano “*Hobby Dev - Free*”, para prosseguir, clicar no botão “*Submit Order Form*”, conforme a Figura 6.

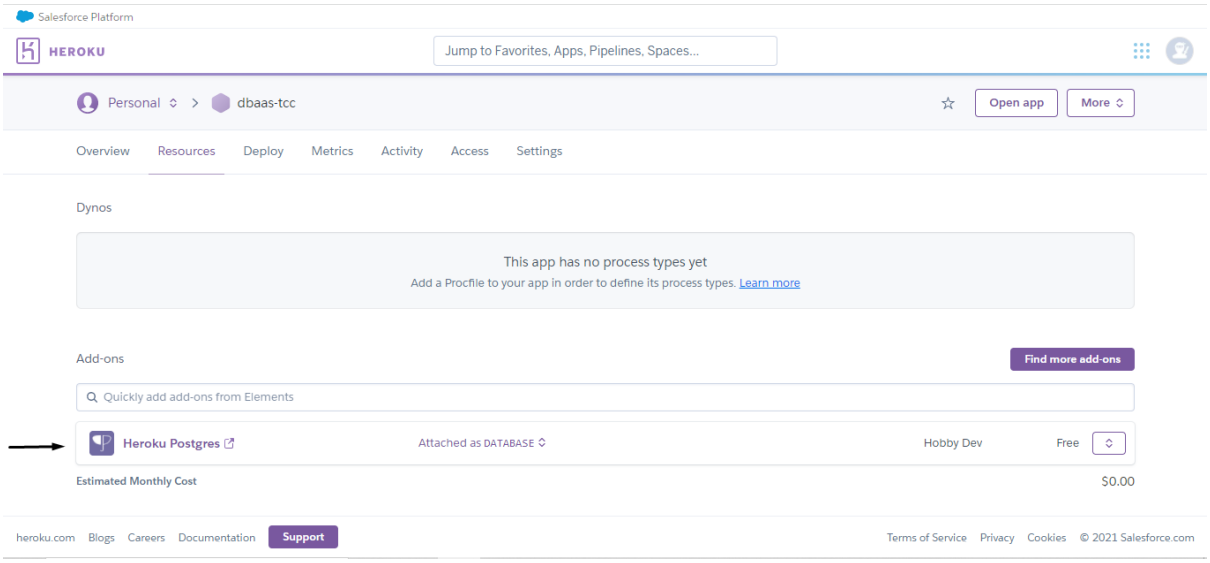
Figura 6 - Escolhendo o tipo de contratação.



Fonte: print screen da plataforma Heroku.

Com isso foi criado o DBaaS, conforme evidenciado na Figura 7.

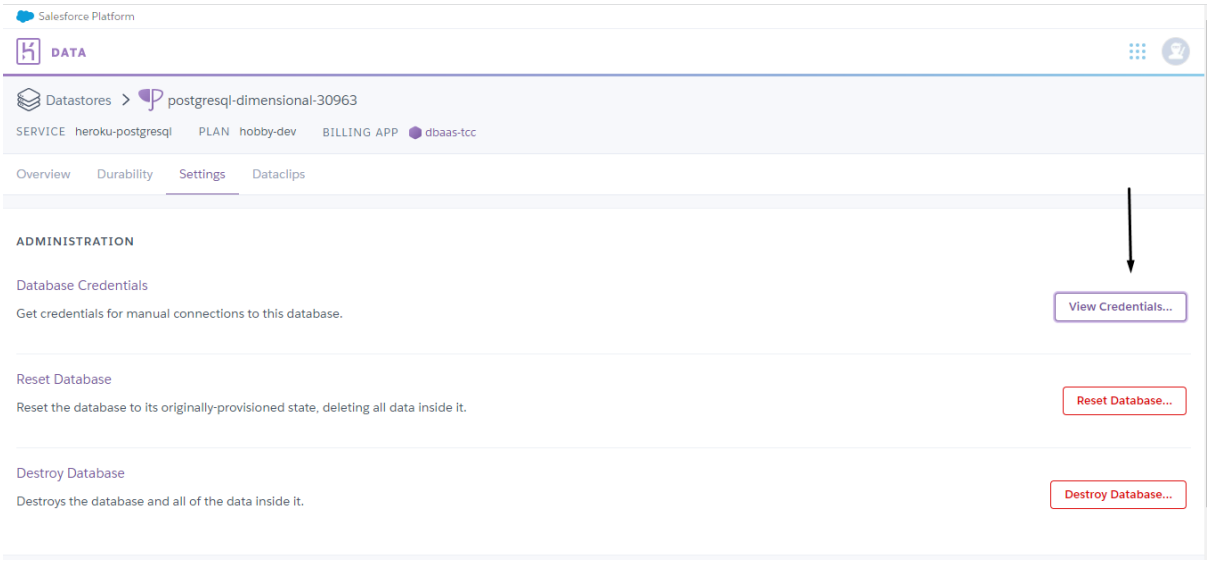
Figura 7 - DBaaS criado.



Fonte: print screen da plataforma Heroku.

Para ter acesso às credenciais de acesso ao DBaaS, clicar no *link* evidenciado na Figura 7. Clicando no *link*, o usuário será direcionado para outra página, conforme Figura 8.

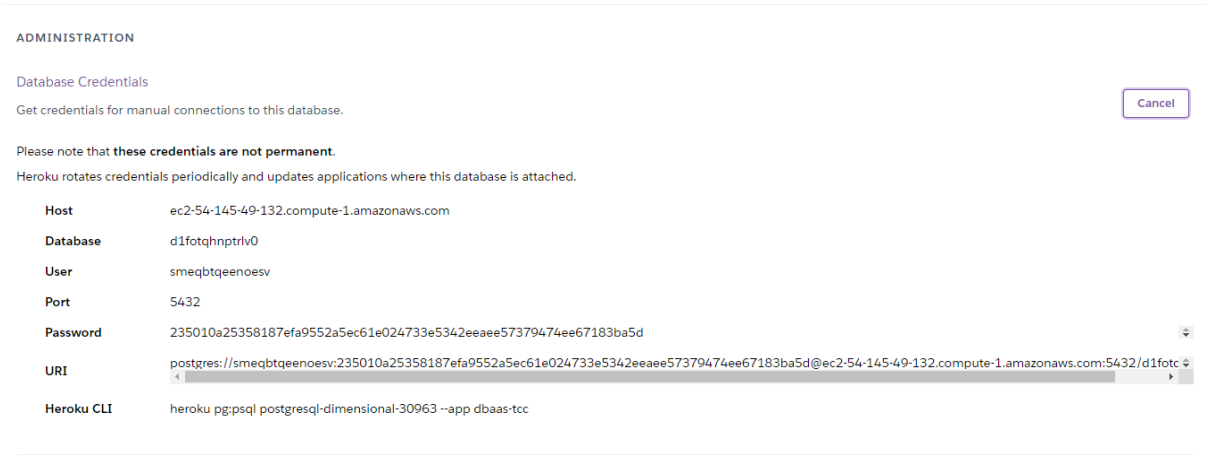
Figura 8 - Página de configuração.



Fonte: print screen da plataforma Heroku.

Clicando na aba "*setting*" e depois no botão "*View Credentials...*" o usuário poderá visualizar as credenciais de acesso ao DBaaS. Nessa demonstração foi gerada as seguintes credenciais evidenciadas na Figura 9.

Figura 9 - Credenciais de acesso ao DBaaS.



Fonte: print screen da plataforma Heroku.

Credenciais geradas para acesso ao DBaaS:

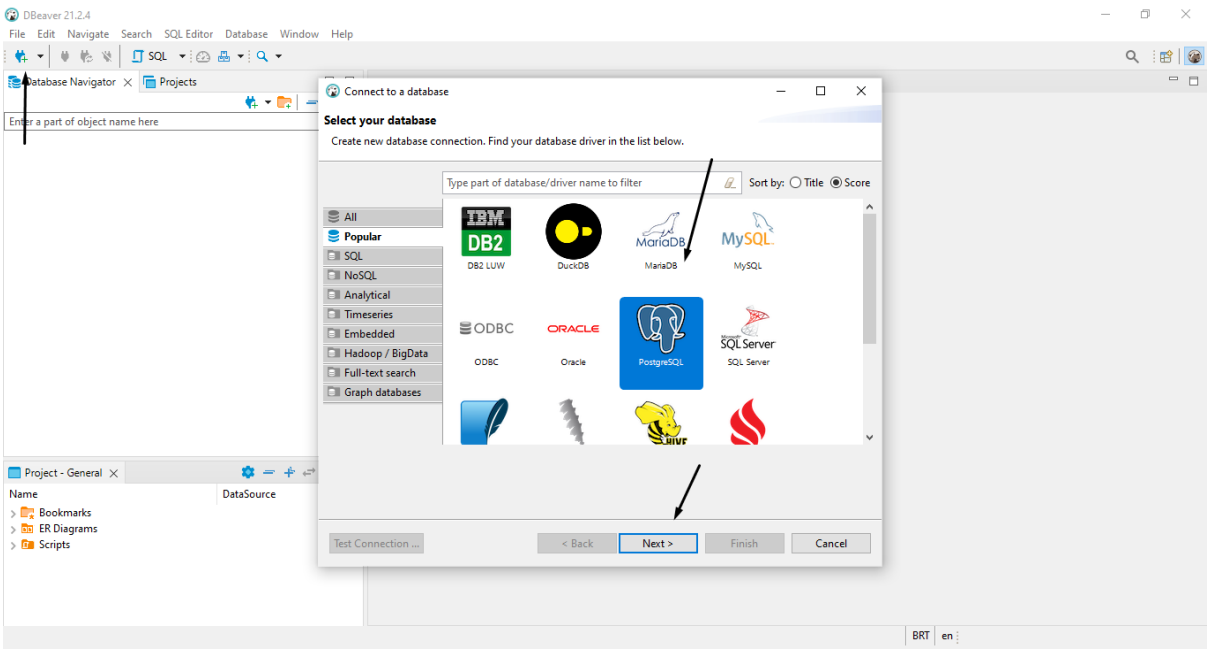
- *Host*: ec2-54-145-49-132.compute-1.amazonaws.com
- *Database*: *d1fotqhnptrlv0*
- *User*: *smeqbtqeenoesv*
- *Port*: *5432*
- *Password*:*235010a25358187efa9552a5ec61e024733e5342eeaee57379474e e67183ba5d*
- *URI*:*postgres://smeqbtqeenoesv:235010a25358187efa9552a5ec61e024733e5 342eeaee57379474ee67183ba5d@ec2-54-145-49-132.compute-1.amazonaws.com:5432/d1fotqhnptrlv*0
- *Heroku CLI*: *heroku pg:psql postgresql-dimensional-30963 --app dbaas-tcc*

7.8 Conexão com o DBaaS

Para acessar o DBaaS criado, será utilizada a interface gráfica *DBeaver*. Com ela é possível acessar o DBaaS *Heroku Postgres*.

Ao abrir a interface, clicar em "*New Create Connection*" e em seguida escolher qual o SGBD, nesse caso o *Postgres*, depois clicar em "*Next*", conforme Figura 10.

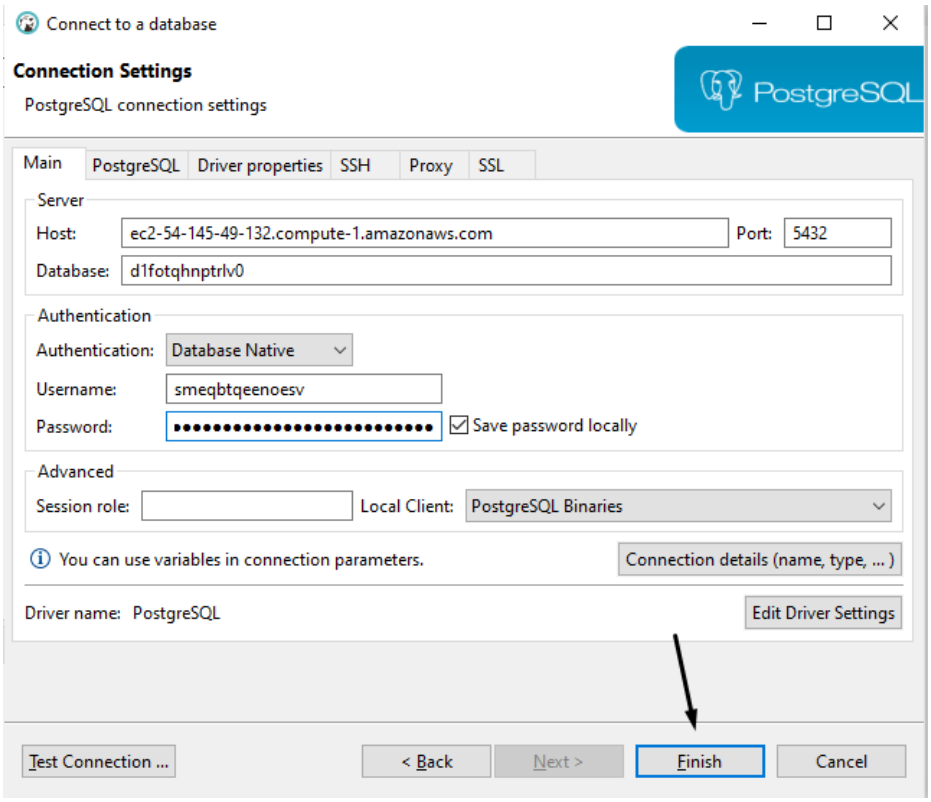
Figura 10 - Escolhendo o SGBD do Postgres.



Fonte: print screen da interface DBeaver.

Após isso será aberta a janela para inserir as credenciais de acesso. Após inserir as credenciais clicar no botão "*Finish*", conforme Figura 11.

Figura 11 - Inserindo as credenciais de acesso.

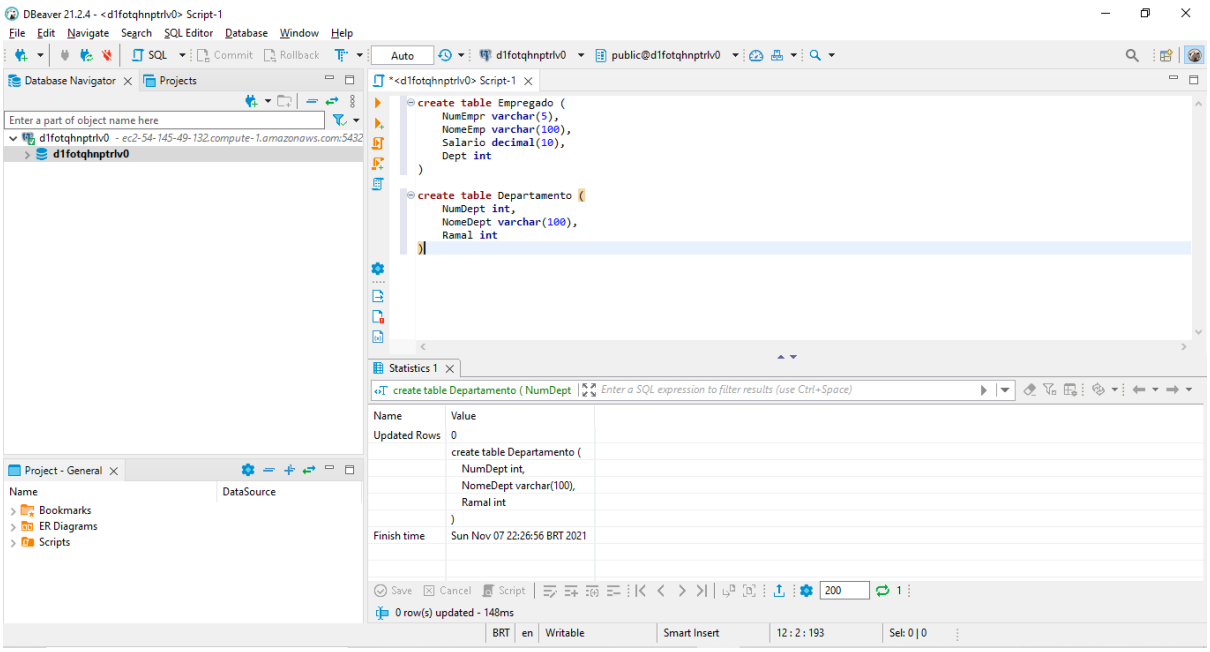


Fonte: print screen da interface DBeaver.

7.9 Criação de tabelas no DBaaS

Para demonstrar a utilização do DBaaS serão criadas algumas tabelas através de comandos DDL. Para executar comandos SQL no *DBeaver* basta clicar na tecla “F3”, após escrever o comando, selecionar os comandos e clicar nas teclas “*Ctrl*” + “*Enter*”.

Figura 12 - Comandos DDL.



Fonte: print screen da interface DBeaver.

O código utilizado para criar as tabelas foi:

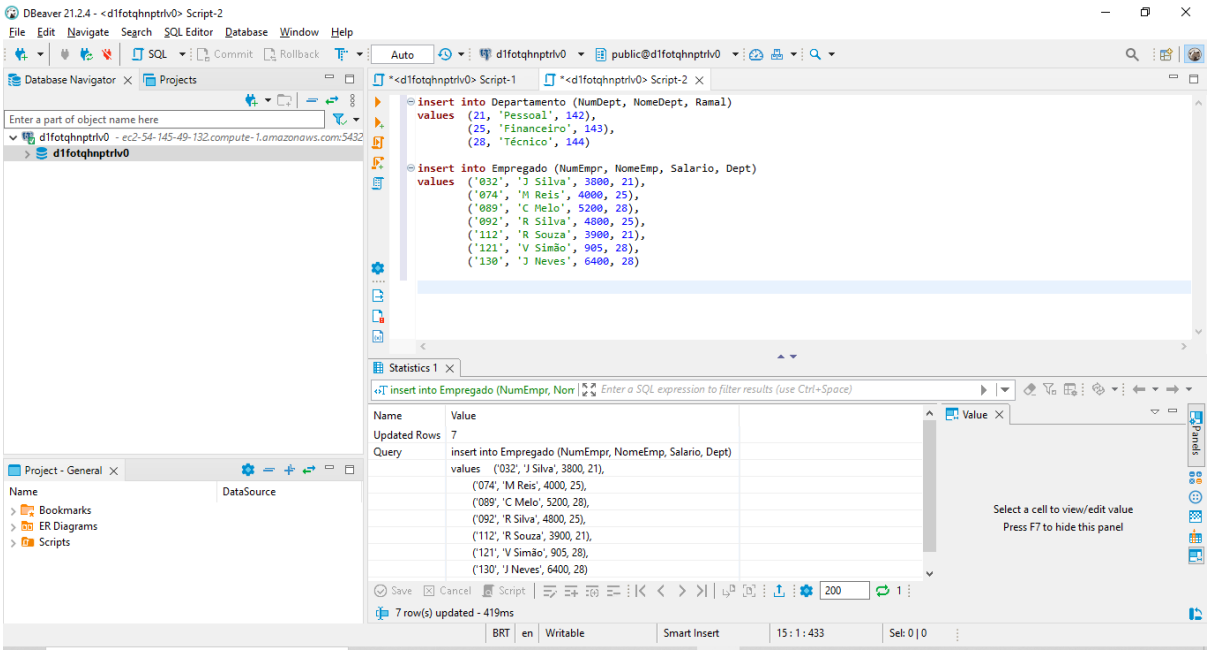
create table Empregado (
NumEmpr varchar(5),
NomeEmp varchar(100),
Salario decimal(10),
Dept int)

create table Departamento (
NumDept int,
NomeDept varchar(100),
Ramal int)

7.10 Inserção de registros nas tabelas

Para inserir registro na tabela será utilizado comando DML, conforme a Figura 13.

Figura 13 - Comandos DML.



Fonte: print screen da interface DBeaver.

O código utilizado foi:

```
insert into Departamento (NumDept, NomeDept, Ramal)
values (21, 'Pessoal', 142),
       (25, 'Financeiro', 143),
       (28, 'Técnico', 144)

insert into Empregado (NumEmpr, NomeEmp, Salario, Dept)
values ('032', 'J Silva', 3800, 21),
       ('074', 'M Reis', 4000, 25),
       ('089', 'C Melo', 5200, 28),
       ('092', 'R Silva', 4800, 25),
       ('112', 'R Souza', 3900, 21),
       ('121', 'V Simão', 905, 28),
       ('130', 'J Neves', 6400, 28)
```

7.11 Extração de dados

Para demonstrar a extração de dados foi utilizado o código a seguir:

```
select d.NomeDept as Departamento , count(e.Dept) as Quantidade, sum(e.Salario)
as Total from Departamento d
inner join Empregado e on e.Dept = d.NumDept
group by d.NumDept, d.NomeDept
```

Esse código extrai a quantidade de empregados e o somatório dos salários de cada departamento. A Figura 14 representa o resultado da extração.

Figura 14 - Resultado da extração.

	123 numdept	ABC nomedept	123 count	123 sum
1	21	Pessoal	2	7,700
2	25	Financeiro	2	8,800
3	28	Técnico	3	12,505

Fonte: print screen da interface DBeaver.

8 CONCLUSÃO

A tecnologia de Banco de Dados como Serviço permite uma maior flexibilidade na forma como o desenvolvedor implementa o Banco de Dados, facilitando a entrega em um ambiente pronto para ser consumido e isso reduz, e muito, o tempo entre o desenvolvimento de uma aplicação e a sua implantação. Essa vantagem é um grande diferencial no momento da escolha da hospedagem, uma vez que o tempo é um recurso valioso para qualquer organização.

O investimento que uma empresa destinaria a infraestrutura de Banco de Dados, pode ser redirecionado para o provedor de DBaaS. Da mesma forma, também não seria necessário a contratação de serviço especializado para o gerenciamento do Banco de Dados, uma vez que as configurações podem ser facilmente alteradas pelo usuário, a escalabilidade pode ser feita de forma automática, assim como a rotina de *backup*.

DBaaS já está quase em fase de maturação tecnológica e com isso pode-se esperar que os pontos negativos, que foram abordados no capítulo anterior, sejam resolvidos ou minimizados nos próximos anos, visto que a maior incerteza gerada em torno dessa tecnologia é a questão da segurança, porém a criptografia avançada permite que os dados sejam armazenados por terceiros de forma totalmente codificada.

Através do estudo de caso foi possível constatar a praticidade em utilizar um DBaaS. Em poucos minutos já havia um Banco de Dados na Nuvem pronto para ser utilizado, bastando somente implementar os objetos e inserir os dados. Isso é uma grande vantagem tanto para quem utiliza de forma profissional, quanto para quem utiliza para fins acadêmicos. E a plataforma da *Heroku* se mostrou bem intuitiva e abrangente, com a possibilidade de criar um DBaaS de forma gratuita e sem burocracia, o que viabilizou a sua utilização.

Com isso pode-se afirmar que este trabalho contribuiu para a compreensão de alguns conceitos fundamentais de Banco de Dados como Serviço e permitiu a demonstração de como a tecnologia funciona na prática, de forma totalmente simples. Também foi possível conhecer a evolução da tecnologia de Banco de Dados, assim como os paradigmas da computação em nuvem. Com isso foram esclarecidas dúvidas e estabelecidos novos conhecimentos a respeito da tecnologia de DBaaS.

REFERÊNCIAS

BARCELAR, R. R. **BANCO DE DADOS**: Introdução ao estudo de bancos de dados. 2012

BERG, K.; SEYMOUR, T. J.; GOEL, R. **History Of Databases**. International Journal of Management & Information Systems. 2012. Disponível em: <https://www.researchgate.net/publication/298332910_History_Of_Databases>. Acesso em: 01 dez. 2021.

CHEN, P. **The Entity-Relationship Model-Toward a Unified View of Data**. Massachusetts Institute of Technology. 1976. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.526.369&rep=rep1&type=pdf>. Acesso em: 01 dez. 2021.

CODD, E. F. **A Relational Model of Data for Large Shared Data Banks**. 1970. Disponível em: <https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf>. Acesso em: 03 de out.de 2021.

CUPPETT, M. S. **DevOps, DBAs, and DBaaS**: Managing Data Platforms to Support Continuous Integration. Tennessee: Apress, 2016.

DATE, C. J. **Introdução a Sistemas de BANCOS DE DADOS**. Tradução de Hélio Auro Gouveia. 7°. Reimpressão. Rio de Janeiro: Editora Campus Ltda, 1984.

ERICKSON, J. Quatro mudanças tecnológicas que estão remodelando o banco de dados corporativo. **Oracle**, 19 de nov. de 2019. Disponível em:<https://blogs.oracle.com/oracle-brasil/post/quatro-mudancas-tecnologicas-que-estao-remodelando-o-banco-de-dados-corporativo>.Acesso em: 10 de out. de 2021.

KOZUBEK, A. The History of SQL Standards. **Learn Sql**, 08 de dez. de 2020. disponível em<https://learnsql.com/blog/history-of-sql-standards>.Acesso em: 17 de out. de 2021.

KRIEGEL, A. **Discovering SQL**: A HANDS-ON GUIDE FOR BEGINNERS. Indianapolis: Wiley Publishing, Inc., 2011.

LOWE, S. D. **Database-as-a-Service**: for dummies. Hoboken: John Wiley & Sons, Inc, 2021.

Magic Quadrant for Cloud Database Management Systems. **Gartner**, 2020. Disponível em:<https://www.gartner.com/doc/reprints?id=1-24BO6U2T&ct=201006&st=sb>.Acesso em: 10 de out. de 2021.

MATOS, D. Top 6 NoSQL Databases. **Ciência e Dados**, 08 de fev. de 2018. Disponível em: <https://www.cienciaedados.com/top-6-nosql-databases/>. Acesso em: 09 de out.de 2021.

MEHAK, F.; MASOOD, R.; GHAZI, Y.; SHIBLI, A.; KHAN, S. Security Aspects of Database-as-a-Service (DBaaS) in Cloud Computing. In: MAHMOOD, Z. **Cloud Computing:** Challenges, Limitations and R&D Solutions. 1° Ed. 2014. p. 297-324.

O que é um Banco de Dados. **Oracle**, 2014. Disponível em:<https://www.oracle.com/br/database/what-is-database>.Acesso em: 12 de out. de 2021.

REDATOR. O que o futuro reserva para o mercado de banco de dados?. **IT Forum**, 30 de ago. de 2019. Disponível em <https://itforum.com.br/noticias/o-que-o-futuro-reserva-para-o-mercado-de-banco-de-dados/>. Acesso em: 10 de out. de 2021.

SETZER, V. Dado, Informação, Conhecimento e Competência. **IME**, 19 de fev. de 1999. Disponível em:<https://www.ime.usp.br/~vwsetzer/datagrama.html>.Acesso em: 01 de dez. de 2021.

SHARMA, S. **Evolution of as-a-Service Era in Cloud**. Center for Survey Statistics and Methodology. 2015. Disponível em: <https://www.researchgate.net/publication/279784427_Evolution_of_as-a-Service_Era_in_Cloud>. Acesso em: 01 de dez. de 2021.

The birth of the IBM PC. **IBM**, 2003. Disponível em: <https://www.ibm.com/ibm/history/exhibits/pc25/pc25_birth.html>. Acesso em: 03 de out. de 2021.

VERAS, M. **Cloud Computing**: Nova Arquitetura da TI. Tijuca: BRASPORT Livros e Multimídia Ltda., 2012.