UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ESPECIALIZAÇÃO EM BANCO DE DADOS

FÁBIO LUIZ UBERTI

COMPARATIVO DE DESEMPENHO ENTRE SISTEMAS GERENCIADORES DE BANCO DE DADOS PARA ARMAZENAMENTO DE DADOS XML EM FORMATO NATIVO

MONOGRAFIA DE ESPECIALIZAÇÃO

PATO BRANCO
2017

FÁBIO LUIZ UBERTI

COMPARATIVO DE DESEMPENHO ENTRE SISTEMAS GERENCIADORES DE BANCO DE DADOS PARA ARMAZENAMENTO DE DADOS XML EM FORMATO NATIVO

Trabalho de Conclusão de Curso, apresentado ao II Curso de Especialização em Banco de Dados, da Universidade Tecnológica Federal do Paraná, campus Pato Branco, como requisito parcial para obtenção do título de Especialista.

Orientador: Prof. Fabiano Luiz Carniel

PATO BRANCO
2017

TERMO DE APROVAÇÃO

COMPARATIVO DE DESEMPENHO ENTRE SISTEMAS GERENCIADORES DE BANCO DE DADOS PARA ARMAZENAMENTO DE DADOS XML EM FORMATO NATIVO.

por

FABIO LUIZ UBERTI

Este Trabalho de Conclusão de Curso foi apresentado em 23 fevereiro de 2017 como requisito parcial para a obtenção do título de Especialista em Banco de Dados. O(a) candidato(a) foi arguido(a) pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Fabiano Carniel
Prof.(a) Orientador(a)

_____
Ives Rene Venturini Pola
Membro titular

_____
Rubia Eliza de Oliveira Schultz Ascari
Membro titular

“O Termo de Aprovação assinado encontra-se na Coordenação do Curso”

À Greice Bombassaro Uberti, pessoa com quem amo partilhar a vida. Com você tenho me sentido completo de verdade. Obrigado pelo carinho, a paciência e por sua importante colaboração.

Agradeço também ao professor Fabiano Luiz Carniel pela orientação e proporcionar conhecimentos imprescindíveis para que eu pudesse alcançar os objetivos deste trabalho.

Se o dinheiro for a sua esperança de independência, você jamais a terá. A única segurança verdadeira consiste numa reserva de sabedoria, de experiência e de competência.

Henry Ford

RESUMO

UBERTI. Fábio Luiz. Comparativo de desempenho entre Sistemas Gerenciadores de Banco de Dados para armazenamento de dados XML em formato nativo. 2017. 44 f. Monografia (II Curso de Especialização em Banco de Dados) - Universidade Tecnológica Federal do Paraná. Pato Branco, 2017.

Os sistemas existentes sejam eles *Desktop*, *WEB* ou *Mobile*, podem utilizar-se de um SGBD (Sistema Gerenciador de Banco de Dados) para armazenar seus dados e agilizar suas consultas, além de manter a consistência dos dados. Portanto, a correta escolha de um SGBD é de suma importância, para que o sistema possa atingir seus objetivos de qualidade, integridade e confiabilidade. Alguns tipos de dados e documentos provenientes de fontes externas precisam ser armazenados em um banco de dados de forma mais eficiente e segura possível. Este é o caso dos arquivos XML (*eXtensible Markup Language*), que são tipos de dados não previstos no modelo relacional de dados, e recentemente suportados de forma nativa nos SGBDs atuais. Nos atuais sistemas existentes no mercado existe uma importante necessidade que é a emissão e recebimento de documentos eletrônicos, sejam eles na área fiscal, saúde, órgãos públicos, entre outros. Documentos estes como, por exemplo: NF-e (Nota Fiscal Eletrônica), TISS (Troca de Informação de Saúde Suplementar), CT-e (Conhecimento de Transporte Eletrônico). Partindo disso, armazenar, validar e posteriormente consultar estes documentos que estão armazenados em seu formato nativo dentro de um SGBD relacional, requer certo cuidado e planejamento na escolha do SGBD, para que atenda da melhor forma possível os requisitos do projeto. Mediante a esse cenário, este trabalho se propõe a definir parâmetros que possam auxiliar na avaliação dos SGBD's. Tendo como foco o armazenamento de documentos do tipo XML e utilizando o *Data Type* XML, ou seja, o documento será armazenado de forma nativa ao banco.

Palavras-chave: Banco de Dados. XML. Documentos eletrônicos. SGBD.

ABSTRACT

UBERTI. Fábio Luiz. Comparison of performance Database Management Systems when storing XML data in a native format. 2017. 43 f. Monography (II Specialization Course in Database) - Federal University of Technology - Parana. Pato Branco, 2017.

Existing systems can be used on Desktop, WEB or Mobile, DBMS (Database Management System) to store your data and streamline your queries, as well as maintain data consistency. Thus, a correct choice of a DBMS is of paramount importance, so that the system can achieve its objectives of quality, integrity and reliability. Some types of data and documents are provided by health care sources and are stored in a database as efficiently and safely as possible. This is the case for XML (eXtensible Markup Language) files, which are data types not foreseen in the relational data model, and recently natively supported in the current DBMSs. In existing systems in the market there is an important need that is an issuance and receipt of electronic documents, they are in the fiscal area, health, public agencies, among others. Documents such as: NF-e (Electronic Invoice), TISS, CT-e (Electronic Transport). From this, storing, validating, and subsequently querying documents that are stored in their native format within a relational DBMS, requires medical care and planning when choosing the DBMS, so that the project requirements are best met. Using this scenario, this paper proposes a definition that can help in the evaluation of the DBMS. XML data type, that is, or document will be stored natively to the database.

Keywords: Database. XML. Electronic documents. DBMS.

LISTA DE SIGLAS

CT-e	Conhecimento de Transporte Eletrônico
DOM	*Document Object Model*
DTD	*Document Type Definition*
ERP	*Enterprise Resource Planning*
HTML	*Hypertext Markup Language*
MSSQL	Microsoft SQL Server
NF-e	Nota Fiscal Eletrônica
RPM	Rotações Por Minuto
SGBD	Sistema Gerenciador de Banco de Dados
SGBDR	Sistema Gerenciador de Banco de Dados Relacional
SO	Sistema Operacional
SQL	*Structured Query Language*
TISS	Troca de Informação de Saúde Suplementar
XML	*eXtensible Markup Language*
XSLT	*eXtensible Stylesheet Language for Transformation*
XSD	*XML Schema Definition*
XPath	*XML Path Language*
XQuery	XML *Query*
WEB	Rede (Referência à rede mundial de computadores)

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta importantes considerações e uma completa visão geral do trabalho, objetivos, justificativa, bem como a organização do trabalho.

1.1 CONSIDERAÇÕES INICIAIS

A tecnologia tem se tornado cada vez mais predominante em todos os segmentos da sociedade. Podemos citar alguns exemplos, como educação, emissão de documentos fiscais, transações bancárias, dentre muitos outros. É fundamental que essa informação seja armazenada, mas também que esteja corretamente armazenada, para que possa ser utilizada com agilidade e eficiência quando houver necessidade.

No Brasil, no que tange a emissão de documentos fiscais, passou-se a utilizar de documentos eletrônicos em formato XML (*eXtensible Markup Language)* para a troca de informações. Documentos estes que podem ser gerados, transmitidos e armazenados pela Receita Federal ou até mesmo pelos contribuintes. Estes documentos XML devem ser armazenados por um determinado período exigido por lei, e esse armazenamento é essencial para que seja possível realizar futuras consultas.

A Linguagem XML foi inicialmente desenvolvida para ser utilizada na *WEB* e foi reconhecida como uma abordagem importante para representar dados semiestruturados. Sua estrutura simples tornou a XML ideal para o intercâmbio de informações entre fontes heterogêneas. Em contrapartida os bancos de dados relacionais representam uma coleção de dados estruturados, organizados em forma de tabelas e registros.

O armazenamento destes documentos, no formato digital em um SGBD (Sistema Gerenciador de Banco de Dados) permite uma maior flexibilidade, segurança e organização. Essa forma torna o processo de armazenamento e consulta destes documentos muito mais fácil e rápido, desde que o SGBD possua o suporte necessário para que seja possível armazenar e consultar os documentos nativamente, ou seja, é uma maneira direta de armazenar os documentos XML em uma coluna, utilizando o *data type* XML, preservando a ordem e estrutura do documento.

Considerando este contexto, torna-se de grande valia definir quais tecnologias em SGBD são mais adequadas para armazenar os documentos eletrônicos em formato XML. Para que esta escolha seja a mais assertiva possível, devem-se estabelecer parâmetros que permitam que o SGBD seja avaliado em uma série de aspectos relevantes, como desempenho, estabilidade, capacidade de processamento.

Esta monografia pretende em linhas gerais, fazer um comparativo entre os principais SGBD's existentes no mercado, para armazenamento de documentos XML em formato nativo, portanto serão considerados os SGBD's: Oracle 11g, Microsoft SQL Server 2016 e PostgreSQL 9.6. Baseado em parâmetros pré-definidos foi possível uma avaliação segura e eficiente no momento da escolha do SGBD a ser utilizado em cada projeto.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Definir parâmetros para que seja possível definir qual SGBD possui um melhor desempenho, para gerir uma quantidade significativa de registros, bem como consultar documentos XML, que foram armazenados utilizando o *Data Type* XML em uma ou mais tabelas do banco de dados.

1.2.2 Objetivos Específicos

- Verificar se as tecnologias propostas armazenam documentos XML em seu formato nativo.
- Definir os parâmetros que devem fazer parte da avaliação do desempenho do SGBD, em rotinas do sistema no momento do armazenamento de campos com *Data Type* XML.
- Avaliar formas de consultas nativas XML.
- Gerar dados comparativos através dos testes efetuados, utilizando os parâmetros estabelecidos.
- Verificar se há como validar o recebimento dos documentos XML ao incluí-los nas bases de dados, utilizando XML *Schema*, por exemplo.

1.3 JUSTIFICATIVA

Esse trabalho justifica-se pela importância que existe na correta definição do SGBD que será utilizado para o armazenamento de dados, pelas aplicações a serem desenvolvidas, como um ERP por exemplo.

É de suma importância que sistemas que irão armazenar documentos do tipo XML sejam adequados, pois isso pode afetar diretamente o desempenho do banco e pode definir o

sucesso ou fracasso do sistema. Este trabalho visa primeiramente definir parâmetros que são características importantes que um SGBD deve conter, para que a aplicação possa atingir seus objetivos com qualidade e desempenho adequado.

Foram delimitados três versões de SGDB`s para abranger esta pesquisa, sendo eles: PostgreSQL 9.6, Oracle 11*g r2* e Microsoft SQL Server 2016. A justificativa pela escolha é pelo motivo de já estarem consolidados no mercado, bem como possuírem uma grande aceitação de seu uso. Foi observado que dos três SGBD's escolhidos, dois são de licenciamento proprietário (Oracle e Microsoft SQL Server) e o terceiro PostgreSQL de código aberto (*Open Source)*.

1.4 ORGANIZAÇÃO DO TEXTO

Este texto está organizado em capítulos, dos quais este é o primeiro e apresenta a ideia e o contexto da pesquisa, incluindo os objetivos e a justificativa.

O Capítulo 2 contém o referencial teórico que fundamenta a proposta conceitual entre os comparativos dos bancos de dados. O referencial teórico está centrado em especificar os parâmetros que serão utilizados como base nas comparações entre os três SGBD's.

No Capítulo 3 estão os materiais e o método utilizados no desenvolvimento deste trabalho. Em Materiais apresenta os bancos pesquisados: Microsoft SQL Server, Oracle 11g e PostgreSQL 9.6. E o Método apresenta a parametrização e a forma comparativa.

O Capítulo 4 contém os resultados obtidos, com em relação a comparação dos critérios usados. E por fim no Capítulo 5 está a conclusão com as considerações finais para esse trabalho.

2 REVISÃO BIBLIOGRÁFICA

2.1 SISTEMAS GERENCIADORES DE BANCOS DE DADOS RELACIONAIS

Segundo o autor Graves (2003), um Banco de Dados é um conjunto de dados armazenados de maneira que persistam e possam ser manipulados. Entendendo que persistência é a permanência dos dados em seus locais depois que o trabalho que os utiliza for encerrado e o computador desligado. Sendo assim a maioria dos arquivos são persistentes, como arquivos textos, planilhas e figuras.

Normalmente quando se fala em um Banco de dados, se estabelece uma aplicação, a qual estará acessando a informação presente no banco de dados, uma camada de acesso e o banco de dados em si, conforme figura 1 a seguir:

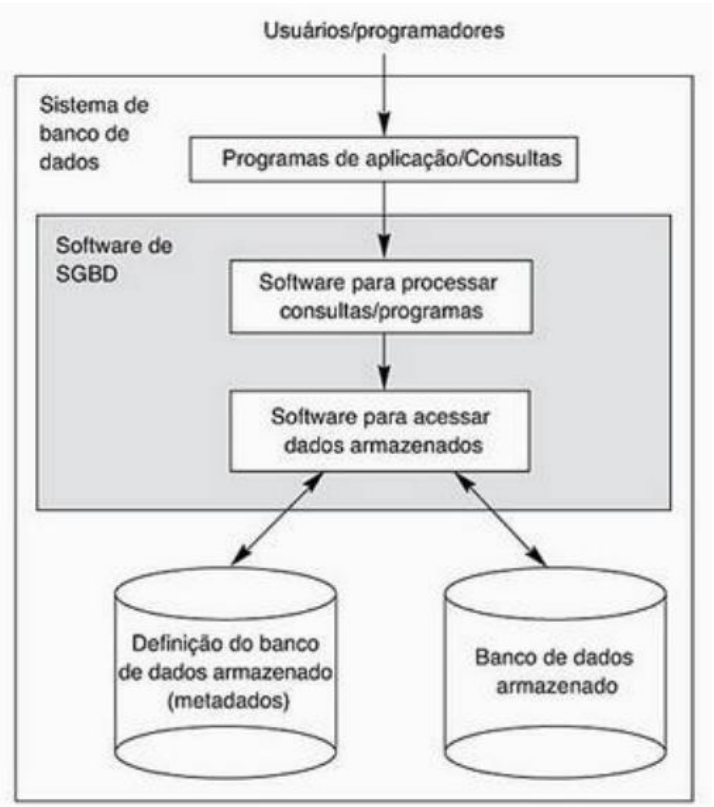

Figura 1 – Diagrama simplificado de um ambiente de banco de dados.
Fonte: Elmasri e Navathe (2011, p. 04)

Observando a Figura 1, é possível verificar que a fonte de alimentação do banco de dados é uma aplicação, sendo a aplicação que está interagindo (usuários) com o mundo real. A camada de acesso por sua vez é quem faz a conexão entre a camada de aplicação e o banco de dados, e na maioria das vezes ela está presente.

De acordo com os autores Elmasri e Navathe (2011), os bancos de dados possuem algumas propriedades, dentre elas é importante destacarmos:

- Um banco de dados representa algum aspecto do mundo real, ou seja, uma parte da rotina de uma empresa, ou de alguma organização, cujos dados precisam ser armazenados. Isso pode ser fisicamente organizado de forma cronológica ou não seu armazenamento.
- Os dados devem se relacionar e seguir uma lógica, dados aleatórios, sem uma lógica na interação não podem ser referenciados como um Banco de Dados. Essa lógica é garantida exatamente pela característica anterior, visto que, se o Banco de dados é baseado em algum aspecto do mundo real, ele segue uma lógica e possui semântica, ou seja, não são meros dados, são dados em um contexto, o que os caracterizam, após alguns processamentos, uma informação armazenada.
- Todo Banco de Dados têm um objetivo específico, portanto, deve ficar claro quais são seus usuários, quais as restrições de acesso, quais as aplicações estabelecidas.

Sendo assim, um Banco de Dados pode ser compreendido segundo os autores Navathe e Elmasri (2011) como sempre sendo uma fonte de alimentação, uma relativa interação com o mundo real, o minimundo e um grupo de usuários. O Banco de Dados ainda possibilita que os dados estejam atualizados a todo instante, ou seja, operações de inserção, remoção e consultas são facilitadas e coordenadas por um software gestor, o SGBD.

O banco de dados relacional é um conjunto de entidades que possuem relacionamentos entre si. Desta forma, o banco de dados relacional preza pela integridade dos dados nele armazenados, não permitindo que os dados sejam armazenados com sua integridade comprometida ou fora de seu devido contexto.

O Sistema Gerenciador de Bancos de Dados (SGBD) é quem coordena o Banco de Dados, e por consequência a sua organização. De acordo com o autor Date (2004), o SGBD é composto por softwares e aplicativos que dinamizam as operações de inserção, consulta e atualização de forma que usuários não autorizados não tenham acesso a componentes críticos do sistema. Assim pode-se compreender que o principal objetivo de um SGBD é proporcionar um ambiente tanto conveniente quanto eficiente para recuperação e armazenamento das informações no Banco de Dados.

Os sistemas gerenciadores de banco de dadas são de suma importância para que um conjunto de dados possa ser armazenado, manipulado e consultado de forma mais simples e objetiva. O SGBDR é que mantém integridade e organização dos dados, para que os mesmo estejam sempre disponíveis e sejam apresentados o mais rápido possível quando solicitados.

2.2 DADOS

De acordo com o autor Rezende (2003) dados são conjuntos de informações puras, podendo ser organizadas ou não. E podem ser considerados como sendo quantitativos, qualitativos, categóricos e indefinidos.

2.2.1 RELACIONAIS

Conforme os autores Navathe e Elmasri (2011), o modelo relacional de dados representa o banco de dados, como sendo uma coleção de relações, cada relação é semelhante a uma tabela de valores ou até certo ponto, um arquivo plano de registros. Ele é chamado arquivo plano porque cada registro tem uma simples estrutura linear ou plana.

Uma relação é considerada uma tabela de valores, onde cada linha representa uma coleção de valores e dados relacionados. Cada uma das linhas representa um fato que normalmente corresponde a uma entidade ou relacionamento do mundo real. (NAVATHE;ELMASRI, 2011).

De acordo com Mello (2003) os Dados Estruturados são dados que seguem um modelo pré-definido, ou seja, na sua representação são regidos por regras, estão subordinados a manter um padrão imposto por um esquema definido antes do conhecimento dos dados.

Segundo os autores Navathe e Elmasri (2011) dados pertencentes ao modelo relacional podem possuir muitas relações em um banco de dados relacional. As tuplas pertencentes a estas relações podem estar relacionadas de várias formas, como por exemplo, *constraints* e outras regras pertencentes ao minimundo do banco de dados.

Sendo assim, os Bancos de Dados Relacionais consistem em um conjunto de tabelas, que contém linhas e colunas e se relacionam entre si, delimitados por uma estrutura rígida. Cada tabela possui várias colunas e cada uma das colunas tem um nome único. Elas possuem ainda colunas que são definidas como campos chaves e possibilitam a conexão entre elas. É possível realizar consultas por meio de linguagens como a SQL (*Structured Query Language*), sendo que toda consulta feita resulta também em uma tabela (NAVATHE;ELMASRI, 2011).

Na Figura 2 pode-se observar um exemplo prático de relacionamento entre as entidades empréstimo, tomador e tom_empr, onde se observa que os dados estão consistentes, devido ao relacionamento e à estrutura rígida proporcionada pelo banco de dados relacional.

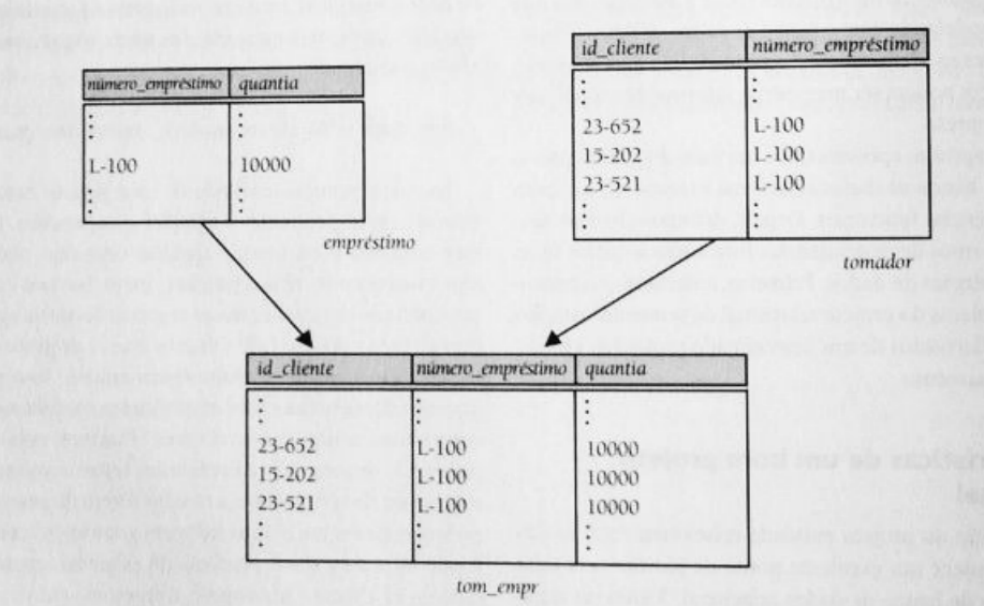

Figura 2 – Lista parcial de tuplas nas relações *emprestimo, tomador* e *tom_empr*.
Fonte: Navathe;Elmasri (2006, p. 176)

Na figura 2, podemos observar um exemplo simples de um relacionamento, em que a entidade tom_empr é a principal do relacionamento. E esta faz a ligação entre as entidades empréstimo e tomador por meio das colunas id_cliente e numero_emprestimo, tornando os dados consistentes, pois estes campos devem possuir estes valores na sua tabela de origem.

Bancos de Dados Relacionais sempre prezam pela consistência dos dados que nele são armazenados, isso se deve ao conjunto de regras de integridade presentes no modelo relacional como integridade de chave e integridade referencial. A organização interna dos SGBDR's é fundamental para que o modelo relacional possa atingir o máximo de confiabilidade, pois sempre que um dado for solicitado este deverá ser localizado e apresentado.

2.2.2 Não Estruturados

Ao contrário dos dados estruturados, os dados não estruturados não apresentam qualquer padrão, não podem ser armazenados de acordo com um esquema. Exemplos de tais dados são: as imagens, o texto livre, vídeos, documentos, dentre outros (MELLO, 2003).

2.2.3 Semiestruturados

De acordo com os autores Abiteboul, Suneman e Suciu (2000) Dados semiestruturados são aqueles nos quais há um sistema de representação presente, explícita ou implicitamente. Pode-se dizer que dados semiestruturados são auto-descritivos.

Os autores Navathe e Elmasri (2011) complementam que em dados semiestruturados as informações do esquema são misturadas com os valores dos dados, pois os objetos podem ter atributos diferentes que geralmente não são conhecidos de forma antecipada. E segundo o autor Mello (2003), ao contrário dos dados estruturados, os dados não estruturados não apresentam qualquer padrão, não podem ser armazenados de acordo com um esquema.

Mesmo em sistemas de integração nos quais as fontes de dados são estruturadas, é comum o aparecimento de informações semiestruturadas. Dados semiestruturados apresentam uma representação de estrutura heterogênea, de forma a não serem completamente não estruturados, nem estritamente tipados (ATZENI; MECCA; MERIALDO, 2016). O exemplo mais prático e conhecido de dados com essa característica heterogênea citada acima, são os dados contidos na web, onde fontes de origens diversas, com padrões diferentes são lançadas neste mundo virtual que é a internet.

Outro ponto que merece ser mencionado em relação a estes dados é que são considerados auto-descritivos, pois possuem uma representação na qual o esquema está presente no próprio dado. Sendo assim, o esquema não é pré-definido, como no caso dos dados estruturados. Esquemas para dados semiestruturados são definidos após a existência dos mesmos e por possuírem esta representação auto-descritiva, os valores e a estrutura em muitos casos se confundem.

2.3 EXTENSIBLE MARKUP LANGUAGE

A XML é uma linguagem de marcação criada com o propósito de estabelecer uma estrutura não tão rígida, mas passível de ser usada para intercâmbio de dados via WEB [W3C, 2017]. Faz-se então necessário entender e visualizar sua estrutura para captar o motivo pelo qual a XML têm-se consolidado por tais características.

XML é uma representação textual de dados. O componente básico em XML é o elemento, ou seja, um texto delimitado por *tags* (marcas), como na Figura 3.

```
<?xml version="1.1" encoding="ISO-8859-1"?>
    <cadastro>
        <nome> Fábio Uberti </nome>
        <dtNascimento> 16/08/1985 </dtNascimento>
        <email>fabio@uberti.com.br </email>
    </cadastro>
```

Figura 3 – Exemplo de documento XML
Fonte: Autor (2017)

A representação textual de dados segue uma sintaxe, o documento deve obedecer as regras abaixo (MARTINS, 2001):

- A primeira linha do documento XML, é opcional e consiste em uma instrução de processamento que define a versão da linguagem XML, define também a codificação usada no documento. No exemplo da figura 3 apresentada anteriormente, a versão do XML é 1.1, usando codificação ISO-8859-1.
- Documentos XML devem conter apenas um elemento raiz, na figura 3, apresentada na página anterior, estará representada por <Cadastro>.
- Cada elemento do documento deve conter um elemento de abertura e outro de fechamento. Por exemplo: <nome> é a *tag* de abertura do elemento e </nome> é a *tag* de fechamento do elemento.
- Deve-se observar e respeitar a ordem de ocorrência dos elementos dentro do documento, ou seja, observar o momento certo de fechar cada *tag*.
- Elementos de abertura e fechamento devem possuir o mesmo nome.
- Cada elemento pode conter um ou mais atributos, cada um com seu respectivo valor entre aspas.
- Cada elemento pode possuir um ou mais sub-elementos.

Com base nesta sintaxe é possível obter o conceito de documento bem-formado é um documento XML, pois atende a sintaxe XML. Se o programador não incluir as marcas de fechamento de um elemento do documento, este documento não será considerado um XML bem-formado (W3C, 2016).

Inicialmente o XML foi criado para que fosse possível transmitir dados na Web uma vez que seu precursor o HTML (*Hypertext Markup Language*) não possuía esta capacidade, pois é muito utilizada para formatação e estruturação de documentos. O XML é uma linguagem de marcação, que tem a capacidade de descrever diversos tipos de dados. O XML tem como principal função ser uma forma fácil de envio e recebimento de informações através da Internet, um exemplo prático de utilização é o envio de documentos fiscais eletrônicos.

2.4 COMPARATIVO ENTRE PARADIGMAS XML E RELACIONAL

A linguagem XML é muito aceita como uma linguagem padrão para intercâmbio entre os meios eletrônicos de dados. Existem alguns fatores muito fortes para a aceitação, dentre estes podemos citar:

> O desenho da XML é formal e conciso, sendo relativamente fácil escrever protocolos que usem seu padrão baseado em XML.
> Documentos XML são facilmente interpretáveis e auto descritivos. Um programador pode abrir um documento XML, lê-lo e facilmente entendê-lo.
> Novas propriedades podem ser facilmente inseridas a qualquer momento. Havendo um grande dinamismo em nível de definição de esquema. (CARNIEL, 2003, p. 37) citação sem os recuos ?

Mesmo o XML possuindo todas as suas facilidades, ainda assim os Bancos de Dados Relacionais são, de fato, o ambiente ideal para o armazenamento eficaz e seguro de dados. Isso se deve, principalmente ao nível de desenvolvimento e maturidade dos bancos de dados relacionais.

A possibilidade de armazenamento de documentos XML na estrutura de um banco de dados relacional, facilita e organiza o armazenamento dos documentos XML. Assim é possível o armazenamento de documentos XML em seu formato nativo e de forma que seja possível manipular estes sem necessidade da existência de dois repositórios distintos para guardar os dados e os documentos XML.

Com o passar do tempo a relação documentos XML e bancos relacionais foi aperfeiçoada, pois como desenvolvimento de linguagens de consulta para documentos XML XPath e XQuery (XML *Query*), estas foram incorporadas aos bancos de dados relacionais. Esta incorporação se deu por meio de plugins, complementos ou de forma nativa, para que seja possível consultar e até alterar documentos armazenados em bancos de dados relacionais.

Dada a grande importância de documentos XML, sendo eles associados a Banco de Dados relacionais, a integração em nível de armazenamento, seleção e atualização de informação tem se tornado de grande necessidade. Deve-se então levantar algumas questões para proceder com integração SGBDR e XML:

- Qual SGBD será utilizado para esta integração?
- Qual o tipo de mapeamento entre XML e o SGBD escolhido?

É válido ressaltar que somente um mapeamento flexível e bem estruturado, a estrutura do documento XML em termos de especificação de esquema é mapeada corretamente para um esquema relacional correspondente, permite realmente explorar a eficiência de gerenciamento de um SGBD Relacional.

A respeito do mapeamento flexível, há problemas quanto à definição do mapeamento entre um esquema XML e um esquema relacional. A heterogeneidade de um modelo de dados se deve ao fato de que há diferenças fundamentais entre os paradigmas XML e relacional. Isso leva por consequência, a concluir que mesmo que um determinado esquema XML e um determinado esquema relacional representem um mesmo universo de estudo, o desenho de

ambos provavelmente será diferente, devido aos objetivos distintos a que foram desenvolvidos.

2.5 ARMAZENAMENTO DE DADOS XML EM BANCOS RELACIONAIS

Os sistemas gerenciadores de bancos de dados, principalmente os bancos de dados relacionais, são atualmente a melhor e mais utilizada forma de armazenamento de dados corporativos, pois fornecem controle de acesso para um grande número de usuários, otimizando as operações e oferecendo segurança a estas informações (GUARDALBEN; SHAKU, 2004).

Bancos de dados relacionais e a linguagem XML possuem aplicações específicas, porém quando integrados constituem uma ótima solução para intercâmbio de dados entre sistemas sejam eles na mesma plataforma ou também em diferentes plataformas. Pode-se observar na Tabela 1 as principais diferenças entre Bancos de Dados Relacionais e documentos XML.

Tabela 1 – Diferenças entre os dados XML e DBR

XML	MODELO RELACIONAL
Armazenamento como arquivo texto.	Armazenamento em ambiente altamente controlado.
Dados em uma única estrutura hierárquica	Dados em múltiplas tabelas
Não contém elementos e/ou valores de atributos	Células contém um único valor
Elementos podem ser aninhados	Valores atômicos em células
Elementos são ordenados	Ordem linha/coluna indefinida
Esquema opcional	Esquema obrigatório
Consulta via padrões XML.	Consultas no padrão SQL

Fonte: Autor (2017)

2.6 TRATAMENTO DE DOCUMENTOS XML NOS SGBD`S

2.6.1 XPath

XPath, a *XML Path Language*, é uma linguagem de consulta (*Query Language*) para selecionar nós em um documento XML. O XPath é usada para consultar valores como por exemplo *strings*, números ou valores booleanos, que estão contidos em um documento XML. XPath foi definido pelo *World Wide Web Consortium* W3C (2017).

Segundo a W3C (2017) XPath é o resultado de esforços para criar uma sintaxe comum e de semântica semelhante ao *XSLT* (*eXtensible Stylesheet Language for Transformation*) e XPoint. Sendo seu objetivo principal poder fornecer facilidades básicas para poder manipular dados como string, numéricos e booleanos.

A linguagem XPath, foi concebida para facilitar a manipulação e consulta de dados em documentos XML. Dessa forma permitindo que sejam construídas instruções XQuery (XML *Query*) semelhante as consultas SQL existentes nos bancos de dados relacionais. Como representado a seguir um comando de consulta XPath para consulta utilizando o cominho completo até o elemento desejado, neste caso o *input[2]*.

Tabela 2 – Exemplo comando XPath

xpath://body/div[3]/form/fieldset/input[2]/ / / /

Fonte: Autor (2017)

2.6.2 PostgreSQL

O PostgreSQL armazena documentos XML usando o *datatype* XML, porém este não dá suporte a validação de *schema* DTD. Esta funcionalidade foi implementada na versão 8.3 com *xmlvalidate*, porém foi considerada uma falha de segurança, uma vez que um usuário sem direito poderia obter arquivos DTD (*Document Type Definition*) de forma arbitrária. Por este motivo foi removida na versão 8.3.1. (COMUNIDADE BRASILEIRA DE POSTGRESQL, 2017)

Existe a possibilidade integrar documentos XML e uma base relacional com o PostgreSQL, criando bases de dados híbridas. Pois existem situações em que o modelo relacional puro seria extremamente ineficiente e uma possível solução utilizando documentos XML dentro da base relacional. Além disso, estão surgindo aplicações de gerenciamento de documentos XML, que podem ter como base de dados um banco relacional. O suporte ao XML no PostgreSQL é feito com uma extensão presente no diretório contrib/xml. É um trabalho baseado em uma biblioteca de funções para manipulação de XML e de consultas XPath (uma linguagem de consulta para documentos XML) já existente chamada libxml. A extensão provê basicamente duas funcionalidades ao PostgreSQL:

- Validar documentos XML com base em um DTD (através da função pgxml_parse).
- Fazer consultas XPath nos documentos armazenados (através da função pgxml_xpath).

2.6.3 Microsoft SQL Server

Microsoft SQL Server fornece uma plataforma avançada para o desenvolvimento de aplicativos avançados para gerenciamento de dados semiestruturados. O suporte para XML é integrado em todos os componentes no SQL Server e inclui o seguinte.

- O tipo de dados XML . Os valores XML podem ser armazenados de modo nativo em uma coluna de tipo de dados XML que pode ser classificada por tipo de acordo com uma coleção de esquemas XML ou deixada sem-tipo. A coluna XML pode ser indexada.
- A capacidade de especificar uma consulta XQuery em dados XML armazenados em colunas e variáveis do tipo XML .
- Aprimoramentos no *openrowset* para permitir carregamento em massa de dados XML.
- A cláusula de FOR XML, para recuperar dados relacionais no formato XML.
- A função OPENXML, para recuperar dados XML no formato relacional.

No MSSQL (Microsoft SQL Server) o tipo de dados XML permite armazenar documentos e fragmentos XML em um banco de dados do SQL Server. Um fragmento XML é uma instância XML que não tem um único elemento de nível superior. É possível criar colunas e variáveis do tipo XML e armazenar instâncias XML nelas.

Opcionalmente, é possível associar uma coleção de esquema XML a uma coluna, um parâmetro ou uma variável do tipo de dados XML. Os esquemas na coleção são usados para validar e classificar as instâncias XML. Nesse caso, diz-se que esse é um XML com tipo.

O tipo de dados XML e os métodos associados ajudam a integrar o XML na estrutura relacional do SQL Server. Para obter mais informações, consulte Métodos de tipo de dados XML.

2.6.4 Oracle

O Oracle Database *8i* introduziu a capacidade de armazenar o código XML no banco de dados, junto com a ampla funcionalidade para manipulá-lo e processá-lo. O Oracle Database *9i* e *10g release 2* adicionaram mais funções de geração de código XML binário em Java e C (o código XML binário fornece armazenamento e manipulação mais eficientes de XML no banco de dados).

O banco de dados Oracle contém várias funções SQL que podem ser usadas para gerar código XML. O Oracle Database 11*g* possui suporte avançado a documentos XML, permitindo que seja possível validar os documentos XML no momento de inserir o

documento XML. Essa validação é realizada com base em XSD e também é validada a construção do XML para que o mesmo seja consistente.

O *datatype XMLType* foi especialmente desenvolvido para seja possível manipular os documentos XML da forma mais nativa possível mesmo estando dentro de um Banco de Dados Relacional .

3. MATERIAIS E MÉTODO

3.1 MATERIAIS

3.1.1 PostgreSQL

O PostgreSQL é um dos resultados de uma ampla evolução que se iniciou com o projeto Ingres, desenvolvido na Universidade de Berkeley, Califórnia. O líder do projeto, Michael Stonebraker, um dos pioneiros dos bancos de dados relacionais, deixou a universidade em 1982 para comercializar o Ingres, porém retornou a ela logo em seguida. Após seu retorno a Berkeley, em 1985, Stonebraker começou um projeto pós-Ingres com o objetivo de resolver problemas com o modelo de banco de dados relacional. O principal problema era a incapacidade do modelo relacional compreender “tipos” (atualmente, chamados de objetos), ou seja, combinações de dados simples que formam uma única unidade. (COMUNIDADE BRASILEIRA DE POSTGRESQL, 2017)

O projeto resultante, chamado Postgres, era orientado a introduzir a menor quantidade possível de funcionalidades para completar o suporte a tipos. Estas funcionalidades incluíam a habilidade de definir tipos, mas também a habilidade de descrever relações - as quais até este momento eram amplamente utilizadas, mas completamente mantidas pelo usuário. No Postgres, o banco de dados "compreendia" as relações e podia obter informações de tabelas relacionadas utilizando regras. (POSTGRESQL FONDATION, 2017)

Em agosto de 1996, Marc Fournier, Bruce Momjian e Vadim B. Mikheev lançaram a primeira versão externa da Universidade de Berkeley e deram início à tarefa de estabilizar o código herdado. Também em 1996, o projeto foi renomeado para PostgreSQL a fim de refletir a nova linguagem de consulta ao banco de dados: SQL. A primeira versão de PostgreSQL, a 6.0, foi liberada em janeiro de 1997. Desde então, um grupo de desenvolvedores e de voluntários de todo o mundo, coordenados pela Internet, têm mantido o software e desenvolvido novas funcionalidades. (POSTGRESQL FONDATION, 2017)

Em maio de 2000 foi liberada a versão 7.0. As versões 7.x trouxeram as seguintes novas funcionalidades: Write-Ahead Log (WAL), esquemas SQL, outer joins, suporte a IPv6, indexação por texto, suporte melhorado a SSL e informações estatísticas do banco de dados. A versão 8.0 foi lançada em janeiro de 2005 e entre outras novidades, foi a primeira a ter suporte nativo para Microsoft Windows (tradicionalmente, o PostgreSQL só rodava de forma nativa em sistemas Unix e, em sistemas Windows - através da biblioteca Cygwin). Dentre as muitas novidades da versão 8.x, pode-se destacar o suporte a tablespaces, savepoints, point-in-time recovery, roles e Two-Phase Commit (2PC). Em setembro de 2010 foi lançada a versão 9.0. (POSTGRESQL FONDATION, 2017)

3.2.1 Microsoft SQL Server

Em 1988, a Microsoft lançou sua primeira versão do SQL Server, ela foi desenvolvida para a plataforma OS/2 juntamente com a Microsoft e a Sybase. Durante os anos 90 a Microsoft iniciou o desenvolvimento de uma versão para a plataforma NT. Enquanto o SQL Server estava sendo desenvolvido, a Microsoft decidiu que ele deveria ser uma camada encapsulada sobre o sistema operacional NT. (MICROSOFT, 2017)

Em 1992, a Microsoft assumiu a responsabilidade maior sobre o futuro do SQL Server para o NT. Em 1993 o Windows NT 3.1 e o SQL Server 4.2 para NT foram lançados. A filosofia da Microsoft em combinar um banco de alta performance com uma interface fácil de usar mostrou-se um sucesso. Microsoft rapidamente tornou-se o segundo mais popular vendedor de softwares de bancos de dados relacionais. (MICROSOFT, 2017)

Em 1994, a Microsoft e a Sybase formalmente encerraram sua parceria, na sequencia em 1995 a Microsoft lançou a versão 6.0 do SQL Server. Esse lançamento foi uma das maiores rescritas da tecnologia SQL Server. A versão 6.0 aumentou a performance substancialmente provendo mecanismos internos de replicação e administração centralizada. (MICROSOFT, 2017)

3.2.3 Oracle

No ano de 1977, ou seja, há quase quarenta anos, Larry Ellison vislumbrou uma oportunidade que outras companhias não haviam percebido, quando encontrou uma descrição de um protótipo funcional de um banco de dados relacional e descobriu que nenhuma empresa tinha se dedicado a comercializar essa tecnologia. Ellison e os co-fundadores da Oracle, Bob Miner e Ed Oates, perceberam que havia um grande potencial de negócios no modelo de

banco de dados relacional, mas não se deram conta de que mudariam a face da computação empresarial para sempre. Sendo assim em 1979 foi lançada a primeira versão comercial de um SGBDR relacional, a versão 2 do banco de dados escrito em linguagem Assembler. (ORACLE, 2017)

Não foi lançada uma versão 1 por medo de as pessoas não comprarem uma primeira versão de software. A primeira versão comercial do software é vendida à Base da Força Aérea americana. Esse é o primeiro SGBDR comercial no mercado, permanecendo no mercado até os dias atuais como um dos maiores bancos de dados relacionais. (ORACLE, 2017)

Após uma serie de inovações e verões, em 1999, foi lançada a verão Oracle 8*i* passa a ter suporte nativo a XML, desta forma foi o pioneiro a possuir este suporte dentre os SGBDR's da época. Na versão seguinte 9*i* e nos posteriores *releases* foi aprimorado o suporte a documentos XML e foram feitas melhorias nas características de tolerância a falhas bem como havendo a cada nova versão melhorias com relação ao desempenho.

3.2 MÉTODO

3.2.1 Definições de Parâmetros para Avaliação de SGBD`S

Após um breve levantamento das funcionalidades dos SGBD`s é possível desenvolver uma tabela comparativa, contendo os parâmetros usados para avaliação dos SGBD's, permitindo que avaliar quais funcionalidades relacionadas ao armazenamento e manipulação de documentos XML cada um deles possui. Podemos observar estas características na Tabela 3:

Tabela 3 – Tabela de Funcionalidades de cada SGBD

Funcionalidades desejáveis	PostgreSQL 9.6	MS SQL Server 2016	Oracle 11*g r2*
Possui tratamento nativo para armazenamento, manipulação e consulta de documentos XML	Sim	Sim	Sim
Necessita de plugins ou add-on para suportar documentos XML	Sim	Não	Não
Permite validação nativa usando *schemas* DTD	Não Possui	Sim	Sim
Documento é validado por um dicionário *XML Schema Definition* (XSD)	Não Possui	Sim	Sim
Possui limite de tamanho de arquivos (SO)	Sim	Não	Não
Qual é tipo de dados para armazenamento XML Nativo em cada SGBD	XML	XML	XMLTYPE
Documento pode ser alterado?	Sim	Sim	Sim
SGBD permite consultas nativas aos documentos XML ? Exemplo: XPath, XQuery	Sim	Sim	Sim

Fonte: Autor (2017)

4 RESULTADOS

4.1 AMBIENTES UTILIZADOS PARA COMPARAÇÃO

Para que fosse possível obter os resultados referentes ao comparativo ente os SGBD's escolhidos, tornou-se muito importante a estruturação de ambientes, para que os testes fossem executados. Visando a praticidade os três ambientes foram montados sob máquinas virtuais utilizando o Oracle Virtualbox 5.

Abaixo está apresentado um breve descritivo de cada ambiente que cada SGBD foi instalado e configurado:

4.1.1 PostgreSQL

Instalação feita em sistema operacional Linux CentOS 6.7. Foi optado por um SO da plataforma Linux, pois segundo a documentação oficial do PostgreSQL, a mesma afirma que pode-se obter uma melhor performance usando esta plataforma. A instalação do SGBD PostgreSQL necessita que a extensão *xml2* seja ativada durante o processo de compilação e instalação, para que o suporte a XML fique disponível.

Na Tabela 4, podem ser observadas quais versões de SGBD e SO foram utilizadas em sua instalação. A instalação foi feita utilizado seu código fonte em sua última versão estável, este obtido diretamente do website do projeto, para a posterior compilação e instalação.

Tabela 4 – Definições do ambiente para o SGBD PostgreSQL .

	Versão
Versão SGBD	PostgreSQL 9.6.1 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.4.7 20120313 (Red Hat 4.4.7-17), 64-bit
Versão SO	CentOS release 6.7
Maquina Virtual	Virtualbox versão 5.1.14r112924

Fonte: Autor (2017)

4.1.2 Oracle

O SGBD Oracle foi instalado em sistema operacional Linux Oracle Linux, usando recomendação da documentação do mesmo, pois é uma distribuição desenvolvida pela Oracle e também é um dos Sistemas Operacionais nos quais o Oracle 11g R2 é homologado. Não foi necessária qualquer configuração especial para que seja possível trabalhar com documentos XML. Na Tabela 5, pode-se encontrar algumas especificações de versões utilizadas para a instalação deste ambiente.

Tabela 5 – Definições do ambiente para o SGBD Oracle.

	Versão
Versão SGBD	Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
Versão SO	Oracle Linux Server release 6.6
Maquina Virtual	Virtualbox versão 5.1.14r112924

Fonte: Autor (2017)

4.1.3 Microsoft SQL Server

Foi instalado em sistema operacional Windows, pois é a única plataforma que até o momento é suportada por este SGBD. Na instalação não foram necessárias quaisquer configurações especiais para que seja possível trabalhar com documentos XML.

A Tabela 6 apresenta as especificações da instalação do Microsoft SQL Server no Microsoft Windows 2012 Server.

Tabela 6 – Definições do ambiente para o SGBD Microsoft SQL Server.

	Versão
Versão SGBD	Microsoft SQL Server 2016 (SP1) (KB3182545) - 13.0.4001.0 (X64) Oct 28 2016 18:17:30 Copyright (c) Microsoft Corporation Enterprise Evaluation Edition (64-bit) on Windows Server 2012 R2 Standard 6.3 <X64> (Build 9600:) (Hypervisor)
Versão SO	Microsoft Windows Server 2012 R2 Standard
Maquina Virtual	Virtualbox versão 5.1.14r112924

Fonte: Autor (2017)

4.2 DEFINIÇÕES DE PARÂMETROS DE AVALIAÇÃO

Para que fosse possível iniciar as rotinas de testes comparativos, inicialmente levantaram-se os requisitos que seriam avaliados em cada SGBD. Na Tabela 7, encontram-se os requisitos que foram utilizados para nortear a avaliação que será exposta no capítulo seguinte.

Tabela 7 – Lista de requisitos a serem avaliados.

Requisito
Armazenar de documento XML
Alterar documento
Verificar se documento XML está bem formado
Validar com DTD e/ou XSD
Consultar dados em documentos XML
Verificar se há diferença de desempenho na inclusão do documento com e sem validação

Fonte: Autor (2017)

4.3 ROTEIRO DE TESTES

Após a definição dos parâmetros a serem avaliados, passou-se a execução das rotinas pertinentes a cada um dos itens elencados na tabela 6. Abaixo estão elencados quais testes foram executados para testar cada uma das funcionalidades.

4.3.1 Criação do Banco de Dados

A criação do banco de dados no Oracle e Microsoft SQL Server 2016, é feita de forma normal, sem que seja requerido qualquer comando ou parâmetro especial para que o mesmo tenha suporte a XML. No entanto o SGBD PostgreSQL requer que seja criada uma *extension* chamada xml2 . Extensão esta que serve para que sejam criadas algumas *functions* nesta base de dados para auxiliar no trabalho com os documentos XML.

4.3.2 Criação de tabelas com campo de tipo XML

No SGBD Microsoft SQL Server 2016 para criar um campo com suporte a XML a sintaxe do campo ***nome_do_campo xml***, como exemplificado no comando da Tabela 7 **.** Caso este campo seja validado por um *schema* então usa-se após o tipo *xml* e na sequencia entre parênteses nome do *schema* no qual será submetida a validação dos documentos XML que serão inseridos. Exemplo de comando ***campo xml (nome_shema_xml)*** como apresentado na Tabela 8. Na Tabela 8 estão representados exemplos do comando de criação.

Tabela 8 – Comandos de criação de tabelas Microsoft SQL Server

Ação	Exemplo
Criar tabela sem validação de *Schema*	*CREATE TABLE nome_tabela (* *coluna_inteiro int primary key,* *coluna_xml xml);*
Criar Schema	*CREATE XML SCHEMA COLLECTION NomeSchemaCollection AS* *N'<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"* *xmlns:tns="http://tempuri.org/PurchaseOrderSchema.xsd"* *targetNamespace="http://tempuri.org/PurchaseOrderSchema.xsd"* *elementFormDefault="qualified">* <xsd:element name='comment' type='xsd:string'/> <xsd:element name='purchaseOrder' type='tns:PurchaseOrderType'/> <xsd:complexType name='USAddress'> <xsd:annotation> <xsd:documentation> Purchase order schema for Example.Microsoft.com. </xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name='name' type='xsd:string'/> <xsd:element name='street' type='xsd:string'/> <xsd:element name='city' type='xsd:string'/> <xsd:element name='state' type='xsd:string'/> <xsd:element name='zip' type='xsd:decimal'/> </xsd:sequence> <xsd:attribute name='country' type='xsd:NMTOKEN' fixed='US'/> </xsd:complexType> </xsd:schema>';
Criar tabela	*CREATE TABLE nome_tabela (* *coluna_inteiro int primary key,* *coluna_xml xml (NomeSchemaCollection));*

Fonte: Autor (2017)

No SGBD Oracle há colunas com suporte a documentos XML, a sintaxe deve ser *campo XMLType*. Em caso de validação de campos utilizando XML *schema* deve-se previamente registrar o *schema* utilizando o comando da Tabela 9, após isso feito pode-se criar o campo usando a seguinte sintaxe. Exemplo:

Tabela 9 – Comandos de criação de tabelas Oracle

Ação	Exemplo
Criar Schema	*XDB.DBMS_XMLSCHEMA.registerSchema('nome_schema', <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://tempuri.org/PurchaseOrderSchema.xsd" targetNamespace="http://tempuri.org/PurchaseOrderSchema.xsd" elementFormDefault="qualified"> <xsd:element name='comment' type='xsd:string'/> <xsd:element name='purchaseOrder' type='tns:PurchaseOrderType'/> <xsd:complexType name='USAddress'> <xsd:annotation> <xsd:documentation> Purchase order schema for Example.Microsoft.com. </xsd:documentation> </xsd:annotation> <xsd:sequence> <xsd:element name='name' type='xsd:string'/> <xsd:element name='street' type='xsd:string'/> <xsd:element name='city' type='xsd:string'/> <xsd:element name='state' type='xsd:string'/> <xsd:element name='zip' type='xsd:decimal'/> </xsd:sequence> <xsd:attribute name='country' type='xsd:NMTOKEN' fixed='US'/> </xsd:complexType> </xsd:schema>')*
Criar tabela	*CREATE TABLE nome_tabela (coluna_inteiro number, coluna_xml XMLType) XMLTYPE COLUMN coluna_xml XMLSCHEMA "*nome_schema*" ELEMENT "elemento_raiz_do_schema";*

Fonte: Autor (2017)

No SGBD PostgresSQL a criação de colunas com suporte a documentos XML possui uma sintaxe simples, bastando apenas informar no nome do campo e o tipo XML e este será criado. No PostgreSQL como não está implementado nativamente um suporte a validação de documentos XML, seja por XML *schema* ou DTD, o tipo XML apenas valida documentos a nível de sua formação, ou seja, se o mesmo é bem formado. Esta validação é feita no momento do *insert* ou *update* do conteúdo no campo.

Para exemplificar como seriam os comandos de criação e tabelas com suporte a documentos XML, na Tabela 10 pode-se ver exemplos de tabelas criadas sem que haja qualquer validação extra, a não ser a validação da formação.

Tabela 10 – Comandos de criação de tabelas

SGBD	Comando SQL de criação da tabela
PostgreSQL	CREATE TABLE PRODUTOS(CODIGO INTEGER NOT NULL PRIMARY, NOME VARCHAR(50), ESPECIFICACOES **XML**);
Oracle	CREATE TABLE PRODUTOS (CODIGO INTEGER NOT NULL PRIMARY KEY, NOME VARCHAR2(50) NOT NULL, ESPECIFICACOES **XMLType**);
Microsoft SQL Server	CREATE TABLE PRODUTOS (CODIGO INTEGER NOT NULL PRIMARY KEY, NOME VARCHAR(50) NOT NULL, ESPECIFICACOES **XML**);

Fonte: Autor (2017)

4.3.3 Inserção de Documento XML

A inserção de documentos XML em campos com suporte a este tipo pode ser feita de forma simples, como demonstrado na Tabela 11, sendo que a inserção se assemelha entre os distintos SGBD's. Esta apresentada na Tabela 11, insere o documento XML em formato de texto plano e apenas validando se a formação do documento XML esta correta.

Tabela 11 – Comandos de inserção

SGBD	Comando SQL de *inserrt* de dados e documento XML
PostgreSQL	INSERT INTO PRODUTOS VALUES (1, 'TECLADO','<ESPECIFICACOES><COR>VERMELHO</COR><TAMANHO>NAO SE APLICA</TAMANHO></ESPECIFICACOES>');
Oracle	INSERT INTO PRODUTOS VALUES (1, 'TECLADO','<ESPECIFICACOES><COR>VERMELHO</COR><TAMANHO>NAO SE APLICA</TAMANHO></ESPECIFICACOES>');
Microsoft SQL Server	INSERT INTO PRODUTOS VALUES (1, 'TECLADO','<ESPECIFICACOES><COR>VERMELHO</COR><TAMANHO>NAO SE APLICA</TAMANHO></ESPECIFICACOES>');

Fonte: Autor (2017)

É possível inserir de outras formas os documentos XML através de repositórios que podem ser criados como no Oracle 11g, pode-se criar um *directory* de onde poderão ser buscados os documentos, porém este não será abordado neste trabalho.

É importante salientar que os três SGBD's estudados já validam a formação do conteúdo XML, no momento da inserção, uma vez que se o conteúdo XML for inválido (por exemplo a falta do fechamento de uma *tag*) a instrução SQL retornará erro e consequentemente será executado o *Rollback* do comando.

No SGBD Microsoft SQL Server e Oracle 11*g* pode-se inserir uma validação baseada em *Schema* XML (XSD), bem como a validação utilizando DTD, por outro lado, esta

validação não está disponível nativamente no PostgreSQL. No PostgreSQL, foram encontradas apenas referências publicadas em fóruns que indicam que esta validação possivelmente possa ser feita através de uma combinação de *functions* e *triggers*.

4.3.4 Alteração de Documento XML

A alteração de documentos XML inseridos em um campo com *Data Type* XML pode ser feita em forma de substituição do conteúdo, ou seja, com um comando de *update* do banco relacional, como demonstrado na Tabela 12, esta funcionalidade está disponível em todos os SGBD's pertencentes a este estudo.

Tabela 12 – Comandos de alteração dos dados do documento XML por *update* relacional

SGBD	Comando SQL de *update* de dados e documento XML
PostgreSQL / Oracle / Microsoft SQL Server	UPDATE PRODUTOS SET ESPECIFICACOES = '<ESPECIFICACOES><COR>AMARELO</COR><TAMANHO>NAO SE APLICA</TAMANHO></ESPECIFICACOES>' WHERE CODIGO = 1;

Fonte: Autor (2017)

Também podem ser feitas alterações por meio de comandos *XPath* como pode ser observado na Tabela 13:

Tabela 13 – Comandos de alteração dos dados do documento XML com *XPath*

SGBD	Comando SQL de *update* de dados e documento XML
PostgreSQL	Nativamente não existe uma forma de alterar apenas um nó do documento XML
Oracle	UPDATE PRODUTOS SET ESPECIFICACOES = UPDATEXML(ESPECIFICACOES,'/ESPECIFICACOES/COR/text()','AMARELO') WHERE CODIGO= 1;
Microsoft SQL Server	UPDATE PRODUTOS SET ESPECIFICACOES = ESPECIFICACOES.modify(replace value of ('/ESPECIFICACOES/COR/ with "AMARELO"') WHERE CODIGO= 1;

Fonte: Autor (2017)

4.3.5 Exclusão de Documento XML

No momento da exclusão de um documento XML, como ele pertence a uma tabela do modelo relacional, ao executar o comando *delete* o registro é eliminado em sua totalidade. A deleção pode ser feita utilizando uma coluna complementar da tabela como filtro para deleção do(s) registro(s) ou caso a tabela possua apenas uma coluna deve-se especificar o documento todo na clausula *where* ou buscando por um atributo especifico usando-se dos comandos *XPath* como condicional. Na Tabela 14, pode-se visualizar os comandos de deleção conforme exposto acima.

Tabela 14 – Comandos de deleção do documento

SGBD	Comando SQL de *delete* de dados e documento XML
PostgreSQL / Oracle / Microsoft SQL Server	DELETE FROM PRODUTOS WHERE CODIGO = 1;

Fonte: Autor (2017)

4.3.6 Consulta de Documento XML

A consulta pode ser feita utilizando-se das funcionalidades disponíveis em cada um dos SGBD's. Segundo a documentação do PostgreSQL, Oracle 11g e Microsoft SQL Server, pode-se observar que todos implementam o padrão de consulta *XPath*, obviamente que em cada SGBD existem peculiaridades em sua implementação e na forma com que podem ser utilizados.

A Tabela 15 expõe exemplos de comandos de consulta, em cada um dos SGBD's utilizando o padrão *XPath*.

Tabela 15 – Comandos de consulta dos dados do documento XML *XPath*

SGBD	Comando SQL de consulta dados e documento XML
PostgreSQL	SELECT XPATH('/ESPECIFICACOES/COR', ESPECIFICACOES) FROM PRODUTOS;
Oracle	SELECT EXTRACT(ESPECIFICACOES, '/ESPECIFICACOES/COR') FROM PRODUTOS;
Microsoft SQL Server	SELECT ESPECIFICACOES.query('/ESPECIFICACOES/COR') FROM PRODUTOS;

Fonte: Autor (2017)

Na Tabela 16, estão demonstrados os comandos para seleção utilizando XQuery, observa-se que o XQuery não está disponível no SGBD PostgreSQL.

Tabela 16 – Comandos de consulta dos dados do documento XML *XQuery*

SGBD	Comando SQL de consulta dados e documento XML
PostgreSQL	Não Implementa XQuery
Oracle	SELECT NOME, XMLQUERY('//ESPECIFICACOES/ESP[@NAME="COR"]' PASSING ESPECIFICACOES RETURNING CONTENT) AS COR FROM PRODUTOS;
Microsoft SQL Server	SELECT ESPECIFICACOES.query('/ESPECIFICACOES/COR') FROM PRODUTOS; SELECT * FROM PRODUTOS where cast(ESPECIFICACOES.query('/ESPECIFICACOES/ESP[@NAME="COR"]/text()') as varchar(50)) = 'BRANCO';

Fonte: Autor (2017)

Observando o que foi exposto até aqui podemos ver claramente que é possível manipular os documentos XML nos SGBD's estudados. Observando também que cada um possui suas limitações e até não oferecendo suporte à algumas funcionalidades como é o caso do PostgreSQL que não implementa a linguagem *XQuery*.

4.4 RESULTADOS OBTIDOS

Esta monografia apresenta um estudo comparativo entre SGBD's relacionais, após um estudo das tecnologias abordadas neste trabalho, estudo este que envolveu a documentação oficial de cada uma das tecnologias, bem como literaturas específicas. Também foram executadas rotinas simples de inserção, alteração e deleção, alguns aspectos puderam ser observados, abaixo seguem aspectos importantes que puderam ser observados e confirmados na documentação oficial de cada SGBD.

- A inserção de documentos XML é semelhante em todos os SGBD's elencados neste trabalho, lembrando de que a inserção é feita de modo simples, ou seja, deve ser inserido um texto contendo o documento XML, desde que contenha todas as *tags* escritas de forma correta. Pois quando é feita a tentativa de inserir um documento inválido um erro é apresentado, informando que aquele documento inserido é inválido, isso se deve ao SGBD's validarem a estrutura do XML na sua formação.
- Algumas documentações, como a do PostgreSQL são de fato muito simples, expondo apenas as funcionalidades básicas referentes ao tratamento de documentos XML, tornando a tarefa de assimilar o seu funcionamento mais complexa, para que seja possível por em prática os conceitos por ela descritos.

- A documentações oficial do PostgreSQL encontrada está obsoleta, mesmo em versões recentes, uma vez que funcionalidades contidas na documentação, já não existem mais no SGBD.

Os três SGBD's estudados, se mostraram aptos a trabalhar com documentos XML. A principal diferença entre eles está relacionada a gama de funções e pacotes de funcionalidades que cada um oferece, fator este que contribui para a sua escolha.

Na Tabela 17, é apresentado um comparativo de funcionalidades e de validações que cada banco possui e também de suporte a linguagens de consulta.

Tabela 17 – Tabela comparativa de funcionalidades

Rotina	Funcionalidade	PostgreSQL	Microsoft SQL Server	Oracle 11g
Insersão/Atualização	Valida Formação nativa	Sim	Sim	Sim
Insersão/Atualização	Valida com DTD nativo	Não	Sim	Sim
Insersão/Atualização	Valida XSD nativo	Não	Sim	Sim
Consulta	XPath	Sim	Sim	Sim
Consulta	XQuery	Não	Sim	Sim

Fonte: Autor (2017)

Observando a Tabela 17, fica claro que dentre os SGBD's estudados, o PostgreSQL é o mais incipiente, pois o mesmo não possui nativamente funcionalidades importantes como validação de documentos baseada em XML *schema* e consultas com XQuery. Enquanto o Microsoft SQL Server 2016 e Oracle *11g* estão mais avançados, desta forma podendo qualificá-los como mais aptos a armazenar e principalmente consultar documentos XML.

Partindo em direção a questão de desempenho que é a questão motivadora deste trabalho, nas tabelas seguintes são apresentadas algumas comparações executadas em cada um dos SGBD's com seus tempos respectivos expressos em milissegundos. Estes tempos podem sofrer variação, conforme a infraestrutura na qual o SGBD se encontra, podendo influenciar a sua performance, um fator que pode ser citado é o *hardware* .

Para a obtenção dos tempos descritos abaixo, foram utilizadas três máquinas virtuais, executadas de forma não simultânea, sobre o sistema operacional Microsoft Windows 8.1 64bits e o *hardware* utilizado foi um processador Intel Core i7 2.20GHz com 16 Gigabytes de memória RAM, disco rígido de 1 Terrabyte com rotação de 5400 RPM.

Para a execução dos comandos pertinentes aos testes foram utilizados três aplicativos *client* para execução dos comandos, são eles pgAdmin III para PostgreSQL, SQL Developer para Oracle 11g e Microsoft SQL Server 2016 Management Studio para o Microsoft SQL Server 2016, para que fosse possível tomar nota dos tempos de execução de cada comando executado abaixo.

Na tabela 18 estão listados os tempos de inserção de um registro e na Figura 4 a sua representação gráfica. Esta inserção contém apenas dois campos na tabela sendo um campo inteiro e outro com *data type* XML. O documento XML continha apenas um nó raiz, dois sub-nós e mais dois tipos complexos, sendo que o segundo nó contém 250 elementos.

Tabela 18 – Inserção de registro simples contendo documento XML.

Comando	Tempo(ms)
PostgreSQL	44
Microsoft SQL Server	2
Oracle 11g	10

Fonte: Autor (2017)

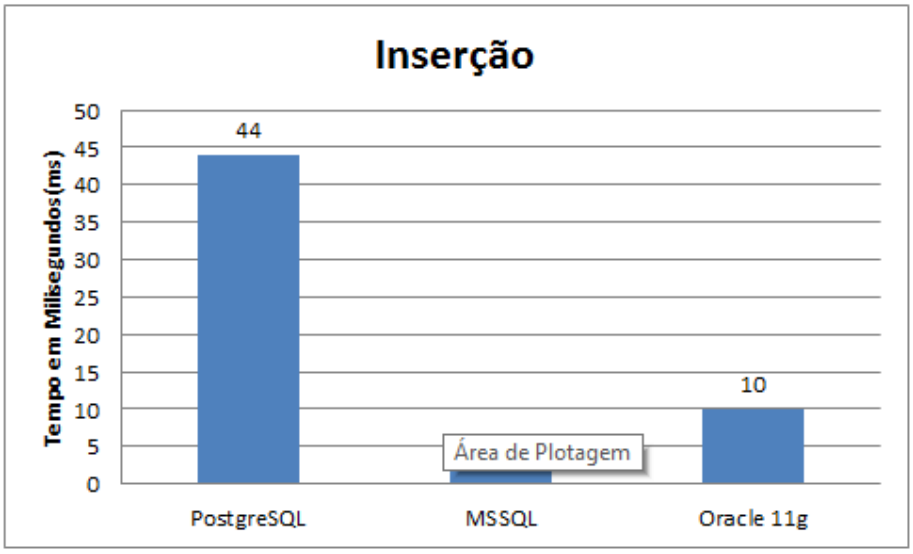

Figura 4 – Inserindo documento XML
Fonte: Autor (2017)

A Tabela 19, apresenta os tempos necessários em cada um dos SGBD's para alterar o documento XML contido em um dos registros e na Figura 5 a sua representação gráfica. A alteração não é um simples *update* e sim a coluna que contém o documento XML é alterada por meio de um comando misto de SQL e XQuery, adicionando um **item** ao tipo complexo **produto** do documento XML.

Tabela 19 – Alteração de um documento XML contido em um registro com XQuery.

Comando	Tempo(ms)
PostgreSQL	Não possui suporte a XQuery
Microsoft SQL Server	6
Oracle 11g	45

Fonte: Autor (2017)

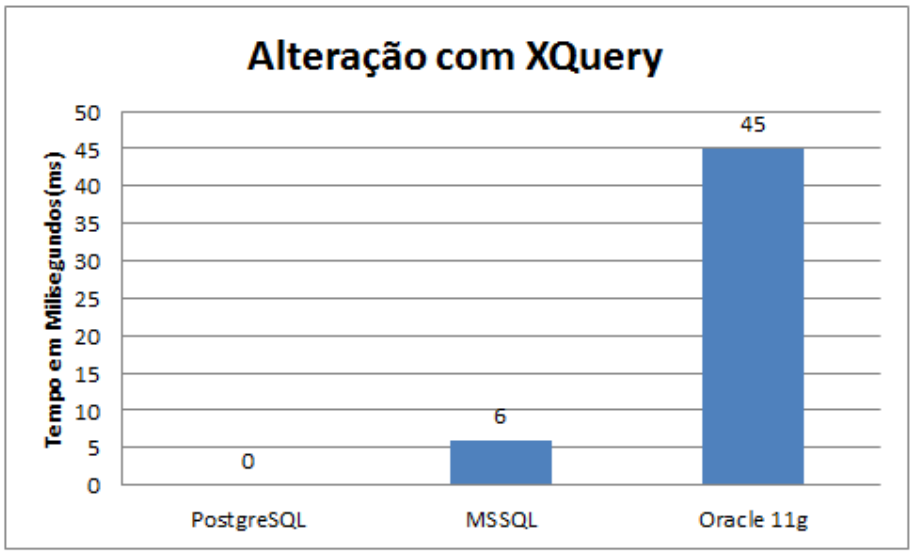

Figura 5 – Alteração de documento XML com XQuery
Fonte: Autor (2017)

A Tabela 20 apresenta os tempos para a mesma alteração e na Figura 6 a sua representação gráfica, porém nesta utilizou-se de SQL e XPath para efetuar a alteração no documento XML.

Tabela 20 – Alteração de um documento XML contido em um registro XPath.

Comando	Tempo(ms)
PostgreSQL	42
Microsoft SQL Server	5
Oracle 11g	10

Fonte: Autor (2017)

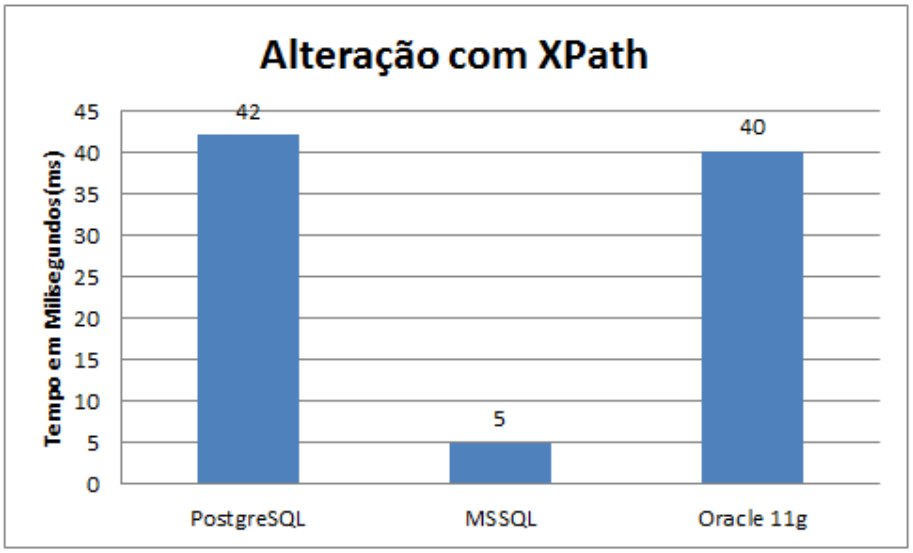

Figura 6 – Alteração de documento XML com XPath
Fonte: Autor (2017)

A Tabela 21 e a Figura 7 representam o tempo necessário para apagar um registro, é importante salientar que a deleção de um registro trata-se de uma operação puramente relacional. Pois quando um nó é excluído de um documento XML em um campo com data type XML, trata-se de uma alteração e não de uma exclusão.

Tabela 21 – Exclusão de registro simples contendo documento XML.

Comando	Tempo(ms)
PostgreSQL	120
Microsoft SQL Server	64
Oracle 11g	60

Fonte: Autor (2017)

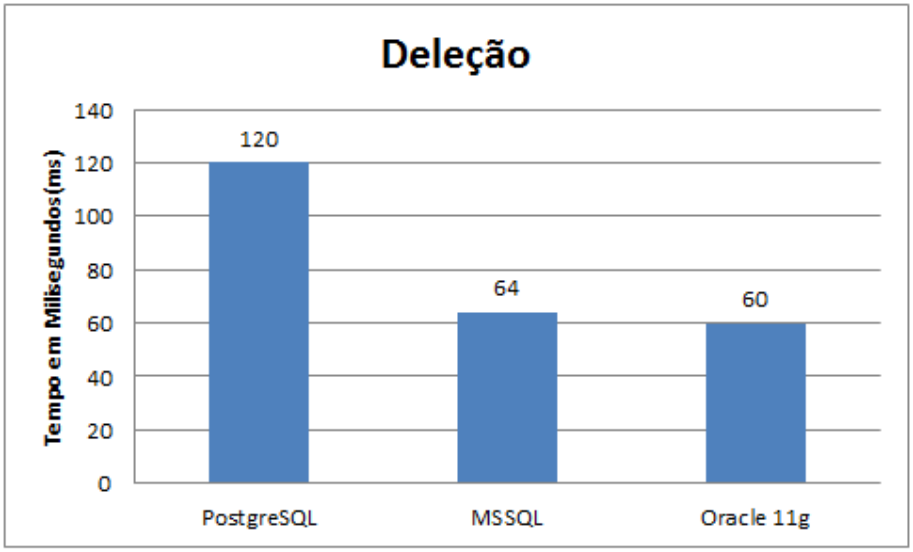

Figura 7 – Excluindo registro com documento XML
Fonte: Autor (2017)

Um fator igualmente ou mais importante que armazenar documentos XML em um banco de dados relacional é a sua recuperação. Os bancos relacionais com suporte nativo a XML, dispõe de linguagens que podem ser utilizadas para efetuar consultas, literalmente navegando entres os nós de um documento como é o caso da XPath.

Está disponível em dois dos três SGBD's estudados a XQuery que é uma linguagem de consulta de documentos XML com maiores possibilidades, principalmente quando se fala de consulta, permitindo efetuar consultas mais complexas. Nas tabelas comparativas Tabela 22 e Tabela 23, e na Figura 8 e Figura 9, estão dispostos os tempos consultas, buscando por documentos XML que contenham o produto com o prod_id = "19".

Tabela 22 – Consulta de documento XML na coluna catalogo usando XQuery.

Comando	Tempo(ms)
PostgreSQL	Não possui suporte a XQuery
Microsoft SQL Server	54
Oracle 11g	60

Fonte: Autor (2017)

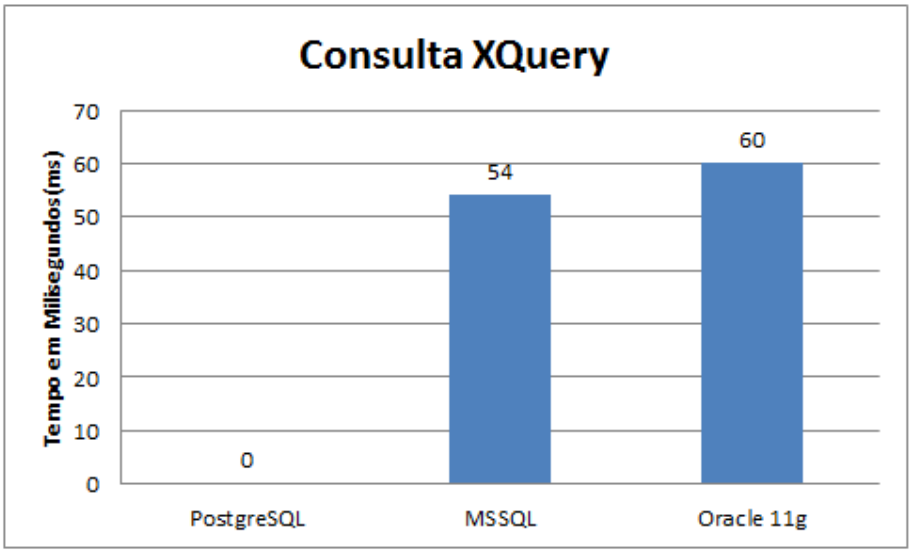

Figura 8 – Consultado documento XML com XQuery
Fonte: Autor (2017)

Tabela 23 – Consulta de documento XML na coluna catalogo usando XPath.

Comando	Tempo(ms)
PostgreSQL	64
Microsoft SQL Server	54
Oracle 11g	60

Fonte: Autor (2017)

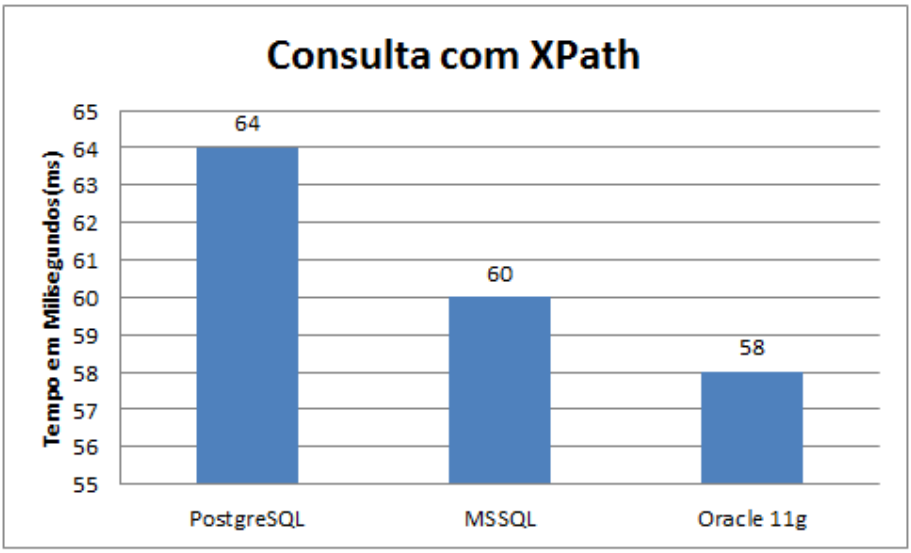

Figura 9 – Consultando documento XML com XPath
Fonte: Autor (2017)

Esta consulta foi executada em uma base que contém dentre outras tabelas, uma tabela chamada fornecedores_catalogo. Nesta tabela existem dois campos o fornec_codigo e catalogo, sendo o campo catálogo do tipo XML. Existe nesta tabela um registro para cada fornecedor e cada fornecedor possui seu catalogo de produtos em formato XML. Estão cadastrados nesta tabela um total de 5000 produtos distribuídos entre os 17 fornecedores cadastrados.

5 CONCLUSÃO

Esta monografia apresenta um estudo comparativo, entre SGBD's relacionais para verificar qual é o mais apto a função de armazenar documentos XML. Este trabalho foi motivado pela necessidade da correta escolha de um banco de dados relacional para armazenar de forma nativa documentos XML, uma vez que os documentos XML são importantes meios de intercâmbio entre sistemas no meio eletrônico.

Pode-se observar que as três tecnologias possuem suporte a XML em seu formato nativo, destacando-se o Microsoft SQL Server 2016 e Oracle 11*g* por possuem mais recursos quanto a manipulação e validação de documentos XML. Em contrapartida observou-se que o SGBD PostgreSQL deixou a desejar nestes mesmos requisitos, apesar do bom desempenho em relação a execução de rotinas de inserção e consulta.

Durante o processo de desenvolvimento deste trabalho foi possível observar que existe uma vasta documentação referente ao armazenamento de documentos XML, para os SGBD`s Microsoft SQL Server e Oracle 11*g*, documentação esta escrita de forma concisa e objetiva, permitindo assim um rápido aprendizado e inicio de sua utilização. Fato este que não foi observado no SGBD PostgreSQL, por sua documentação estar incompleta em algumas funcionalidades e desatualizada em outras, dificultando um pouco a utilização.

Esta monografia compara os SGBD's na questão de recursos disponíveis e desempenho de forma a permitir uma escolha mais assertiva de um SGBD para armazenar documentos XML . O estudo e a comparação das soluções implementadas nos SGBD's deste estudo podem servir como base para trabalhos futuros focados no armazenamento de documentos XML em SGBD's relacionais.

REFERÊNCIAS

ABITEBOUL, S. BUNEMAN, P. SUCIU, D. **Data on the Web: from relations to semistructured data and XML**. San Francisco: Morgan Kaufman Publishers, 2000.

ATZENI, P. MECCA, G. MERIALDO, P. **Semistructured e Structured Data in the Web: Going Back and Forth**. Disponível em http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.57.3236&rep=rep1&type=pdf Último acesso em 02/11/2016.

CARNIEL, Fabiano L. **Análise comparativa do mapeamento de dados XML para bancos de dados relacionais.** 2003. 37 f. Monografia (Especialização em Ciência da Computação) - Universidade do Oeste de Santa Catarina (UNOESC) Xanxerê Pró-Reitoria de Pesquisa, Extensão e Pós Graduação Centro de Ciências Sociais Aplicadas, Xanxerê, 2003.

COMUNIDADE BRASILEIRA DE POSTGRESQL. **Documentação**. Disponível em: https://www.postgresql.org.br/docs Último acesso em: 10/01/2017

DATE, C. J. **Introdução a Sistemas de Banco de Dados**. 8 ed. São Paulo: Editora Campus Ltda., 2004.

ELMASRI. R. , NAVATHE. S. B. **Sistemas de banco de dados.** 6 ed. São Paulo: Pearson Education do Brasil, 2011.

GRAVES, Mark. **Projeto de Banco de Dados com XML**. São Paulo: Makron Books, 2003.

GUARDALBEN, G.; SHAKU, A. **Integrating XML and Relational Database Technogies: A Posistion Paper.** In: HIT SOFTWARE. White Papers . Disponível em: <http://www.hitsw.com/products_services/whitepapers/integrating_xml_rdb/integrating _xml_white_paper.pdf>. Último acesso em: 02/11/2016

MARTINS, W. R. **Servidor de documentos XML usando java**. Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação, da Universidade de São Paulo - USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Área de Ciências de Computação e Matemática Computacional. 2001.

MELLO, R. d. S. **Dados semi-estruturados**. https://www.ime.usp.br/~jef/semi-estruturado.pdf Ultimo acesso em 02/11/2016, 2003.

MICROSOFT. **SQL Server Technical Documentation**. Disponível em: https://msdn.microsoft.com/en-us/library/ms130214.aspx Último acesso em: 10/01/2017

ORACLE. **Oracle Database Online Documentation 11g Release 2 (11.2).** Disponível em: http://docs.oracle.com/cd/E11882_01/index.htm Último acesso em: 10/01/2017

ORACLE. **A história do Oracle: Inovação, Liderança e Resultados** www.**oracle**.com/br/corporate/newsroom/story-346137-ptb.html Último acesso em: 10/01/2017

POSTGRESQL FONDATION. The PostgreSQL Global Development Group. **Documentation**. Disponível em: https://www.postgresql.org/docs/ Último acesso em: 10/01/2017

REZENDE, S. O. **Sistemas Inteligentes**. 1 ed. São Paulo:Manole, 2003.

W3C. **Extensible Markup Language (XML)**. Disponível em: https://www.w3.org/XML/ Último acesso em: 02/11/2016.

W3C. **XML Path Language (XPath) Version 1.0**. Disponível em: https://www.w3.org/TR/xpath/ Último acesso em: 09/01/2017.