# COMPARAÇÃO PRÁTICA ENTRE JSON E XML - ESTUDO DE CASO

## *PRACTICAL COMPARISON BETWEEN JSON AND XML - CASE STUDY*

**XXXXXXXX[1], YYYYYYYY[1]**
[1]ZZZZZZZZ
aaaa – bbbbb – bbb
e-mail: ----------

*Abstract*

*The choice between the most appropriate format for representing and exchanging data is of great importance as it impacts on the performance of the various applications that use them. Currently XML and JSON are the most commonly used formats for this purpose. In this article, we will describe the comparison of these data formats in terms of the size of the file to be exchanged. An evaluation is made with the files in their natural state, i.e. without conversion, and compressed, using a public compressor. The data conversion time and the size of the compressed file were compared. The obtained results were the same for the two when compressed 1 and 10 registers, however for number of registers greater than 1000 the XML proves to be smaller than the JSON.*

***Keywords****: XML, JSON, network bandwidth consumption, XML versus JSON compression.*

**Resumo**

A escolha entre o formato mais apropriado para representação e intercâmbio de dados, é de grande importância, pois impacta no desempenho das diversas aplicações de software. Atualmente XML e JSON são os formatos mais usados para este fim. Neste artigo, descreveremos uma comparação entre esses formatos de dados em termos de tamanho do arquivo a ser intercambiado. É feita uma avaliação com os arquivos em seu estado natural, i.e. sem conversão, e comprimido, usando um compressor público. Comparou-se o tempo de conversão de dados e o tamanho do arquivo compactado. Os resultados obtidos foram iguais para os dois tipos de arquivos quando 1 e 10 registros são compactados, porém para quantidade de registros maiores que 1000 o arquivo XML compactado é menor que o JSON, consumindo dessa formar menor banda de rede na transmissão.

**Palavras-chave**: XML, JSON, consumo de banda de rede, compressão XML versus JSON.

## 1. Introdução

No decorrer dos anos a tecnologia tem apresentado grandes avanços e com isso grandes benefícios para a sociedade. Esses benefícios interferem diretamente nas atividades diárias das pessoas e estão tão presentes que muitas vezes passam despercebidas. Como exemplo são os web services que são responsáveis pela integração e intercâmbio de informações entre diversas aplicações [8].

Os web services utilizam linguagens que permitem que aplicações antigas e/ou desenvolvidas em plataformas diferentes se tornem compatíveis através da troca de informações nos mesmos formatos e sintaxes [8]. Entre as vantagens da utilização de web services podemos destacar a viabilização de recursos de diversas aplicações esteja disponibilizada em uma única rede de forma normalizada garantindo mais eficiência na comunicação entre as aplicações de forma dinâmica e segura dispensando a intervenção humana [9]. Atualmente os formatos mais utilizados para a transferência de dados entre aplicações são XML [2] e JSON [4].

Este trabalho surgiu da necessidade de investigar e otimizar a integração de sistemas, e.g. sistemas que utilizam diversas aplicações para a área fiscal, sendo estes utilizados por pessoas jurídicas, pessoas físicas e até órgãos de governo. Para o tráfego dessas informações entre as máquinas dos clientes e os servidores é utilizado o formato de arquivo XML que pode gerar um arquivo final com tamanho que, apesar de pequeno, acaba por se tornar um grande problema quando multiplicado pelo grande fluxo e a grande demanda de entrada nos servidores causando um gargalo em todo o processo.

Outro aspecto que gera mais uma dificuldade refere-se a situações nas quais as conexões com a Internet são bastante limitadas e com velocidades e banda de rede reduzidas, e.g. municípios remotos do território brasileiro, nesses casos, as transferências de arquivos para cumprir requisitos legais é mais outra dificuldade enfrentada pelos usuários.

Este trabalho mensura o tamanho de arquivos em formato XML e JSON com registros extraídos do banco Sakila [11], e compara-os com o tamanho após a conversão utilizando a ferramenta de compactação de arquivos gzip [12]. O ambiente operacional para este estudo de caso é constituído por um programa desenvolvido por Níckolas Daniel [11], em PHP, o qual realiza consultas em uma tabela de filmes e converte para registros em XML e JSON para em seguida compactá-los. Encontramos evidências significativas para inferir que os arquivos XML, com grandes registros, são menores que os JSON, e possuem tempo de compactação mais rápido.

Este artigo está organizado a partir dessa introdução, seguido pela seção 2 que descreve a linguagem XML e JSON, na seção 3 é feita uma discussão sobre trabalhos relacionados, na seção 4 é apresentado o experimento, na seção 5 são discutidos os resultados obtidos, e, por fim, conclui-se na seção 6.

## 2. Conceitos

### 2.1. XML

A Extensible Markup Language (XML) [2] é um subconjunto da Linguagem de Marcação Generalizada Padrão (SGML) [6] e evoluiu em função da complexidade da SGML. A XML é considerada o 'santo graal' da computação devido ao seu formato universal de representação de dados [1]. A intenção de um documento XML é evidente e incorporada em sua estrutura. As considerações fundamentais de design da XML incluem simplicidade e legibilidade humana. Entre os objetivos de design da XML, o W3C especifica que “XML deve ser diretamente utilizável pela Internet” e “os documentos XML devem ser legíveis e razoavelmente claros” [2].

Os principais usos para XML são Remote Procedure Calls (RPC) [3] e serialização de objetos para transferência de dados entre aplicativos. XML é uma linguagem usada para criar marcações definidas pelo usuário para documentos e esquemas de codificação. A XML não tem conjuntos de tags predefinidos e cada tag válida é definida por um usuário ou por outro esquema automatizado. Um grande número de tutoriais e fóruns de usuários fornece amplo suporte para XML e ajudou a criar uma ampla base de usuários. XML é um formato de dados hierárquico definido pelo usuário. Um exemplo de um objeto codificado em XML é fornecido na Figura 1.

```xml
<pessoa>
    <nome>Nawarian Nickolas</nome>
    <idade>20</idade>
</pessoa>
```

**Figura 1: Uma estrutura hierárquica descrevendo a codificação do nome e idade.**

## 2.2. JSON

A linguagem JSON [4] foi projetada para a troca de dados com legibilidade e facilidade de ser analisada e usada por computadores. JSON é suportada diretamente no JavaScript [5] e é mais adequada para aplicativos escritos nessa linguagem (JavaScript); proporcionando assim ganhos significativos de desempenho em relação à XML, o que requer bibliotecas extras para recuperar dados de objetos DOM (Document Object Model) [7]. Calcula-se que, em navegadores modernos, o documento JSON seja analisado até cem vezes mais rápido que o XML [4], mas apesar de suas alegações de desempenho notável, os argumentos contra JSON incluem falta de suporte ao namespace, falta de validação de entrada e desvantagens de extensibilidade.

Crockford [4] aborda esses argumentos afirmando que “todo objeto é um namespace. Seu conjunto de chaves é independente de todos os outros objetos, mesmo exclusivos de aninhamento. Além disso, Crockford [4] afirma que JSON usa o contexto para evitar a ambiguidade, assim como as linguagens de programação”, que a validação de entradas é responsabilidade de aplicativos de domínio individuais e que a falta de solicitações de extensibilidade é abordada pela flexibilidade das construções JSON. Ademais, a sintaxe do JSON é legível por humanos. A Figura 2 ilustra um exemplo em que o JSON é usado para codificar um nome e um sobrenome.

```json
{
    "nome": "Nawarian Nickolas",
    "idade": 20
}
```

**Figura 2: uma construção JSON simples descrevendo a codificação de um nome e da idade**

## 3. Trabalhos relacionados

Várias comparações baseadas em diferentes cenários foram feitas entre os dois formatos [14]. Além disso, uma análise abrangente de XML e JSON para tecnologia web é descrita em [13]. Em [15], a troca de dados de processo entre um aplicativo movel e servidores remotos usando o formato JSON é descrita. Uma comparação de desempenho entre os dois formatos de intercâmbio para estruturas simples é descrita em [16]. Conforme destacado em [17], é possível alternar entre um formato para outro usando conversores,

preservando o conteúdo dos dados. A conclusão comum é que, geralmente, o JSON, com seu formato simples, se comporta mais rapidamente e usa menos recursos comptacionais que XML. Em [18] é apresentada uma comparação de desempenho de benchmark entre os formatos de dados JSON e XML, tanto do ponto de vista do tempo de execução, quanto do uso da memória, usando diferentes tipos de dados. Foram realizados testes de desempenho explorando o tempo de execução e o consumo de memória para métodos de envio de dados para vários aplicativos [18]. Além de outros estudos comparativos existentes na literatura, tentamos considerar em comparação, para os dois formatos o tamanho final após a compressão de dados.

Este artigo difere dos demais pois apresenta uma comparação de desempenho de benchmark entre os formatos de dados JSON e XML compactados. O objetivo principal é avaliar o tamanho do arquivo compactado e o tempo de compactação entre os formatos.

## 4. XML ou JSON para o inetrcâmbio de dados?

Em função da verbosidade da XML, o arquivo JSON costuma ser menor, uma vez que para descrever um trecho de informação necessita de uma quantidade menor de caracteres [4]. Por outro lado, o arquivo XML permite uma melhor estruturação dos dados, utilizando as tags de marcação [3] que não estão presentes em JSON.

Para conversão e compactação de dados em XML e JSON foi utilizado o PHP [19] no lado do servidor. Para podermos analisar os dois formatos foi utilizada as funções criadas por Níckolas Daniel [11]. As funções foram então analisadas com base em 3 critérios: tamanho sem compactação, tempo de compactação e tamanho após a compactação.

O algorítimo monta os arquivos JSON e XML extraindo do banco SAKILA 1 milhão de registros, calculando o tempo de compactação e o tamanho final do arquivo já compactado. Para o análise a que se propõe este artigo foi desconsiderada a metodologia utilizada para gerar os arquivos.

A Figura 1 ilustra a estrutura da tabela *film* do banco de dados SAKILA, que é usada no algorítimo, contendo dados de amostra de filmes. Para o experimento o critério adotado para escolha da tabela foi possuir mais de 1 milhão de registros.

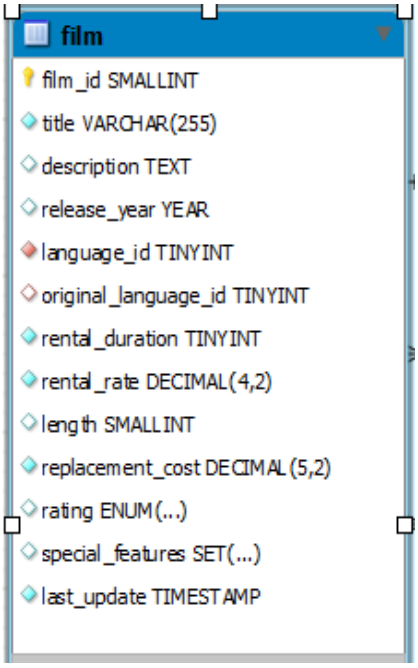

Figura 1. Estrutura da tabela film

Os dados da tabela são manipulados usando consultas SQL conforme mostrado na Figura 2, que traz os registros da tabela *film* do banco SAKILA, e na Figura 3 é mostrado o trecho do código que armazena os dados na variável *$dados*.

```
10
11  return $pdo->query( 'SELECT * FROM film' )->fetchAll( PDO::FETCH_OBJ );
```

Figura 2 – Código de extração de dados da tabela *film*.

```
1  <?php
2
3  $dados = require_once __DIR__ . '/../resources/obter-dados.php';
4
```

Figura 3 – Código de armazenamento de dados da tabela *film* na variável *$dados*.

A Figura 4 contém o trecho do algorítmo em que é feita a compressão do arquivo gerado para os dois formatos. Na linha 4 e 6 do código são armazenados os tempo de compressão, na linha 5 a função gzdeflate [22] do PHP é utilizada para calcular o tamanho comprimido e na linha 7 calcula-se a diferença entre os tempos de compressão.

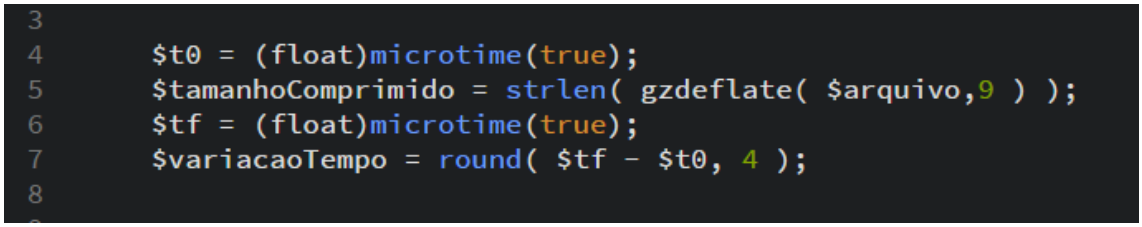


Figura 4 – Algoritmo para compressão.

### 4.1 O Experimento

A proposta deste experimento é comparar o comportamento dos dois formatos, XML e JSON, e tentar identificar qual o mais indicado levando em consideração fatores como o formato do arquivo, tamanho do arquivo compactado e o tempo de compactação.

Sabendo que os arquivos XML tendem a apresentar maior tamanho, nesse experimento foi adicionado um procedimento a ser utilizado antes do envio do arquivo pela rede de intercâmbio de dados, o qual seria compactar os arquivos gerados utilizando o algoritmo Deflate [20].

O algorítimo Deflate [20] utiliza um método de codificação estática, que descobre a probabilidade de cada símbolo ocorrer e os junta sucessivamente com os de menor probabilidade, construindo uma árvore binária, através de estatística, o que possibilita obter códigos binários para cada símbolo [21]. Isto, para nosso experimento, auxiliaria na compressão dos arquivos XML, em função de sua estrutura utilizar esta metodologia.

#### 4.1.1 Ambiente

Para realizar os testes foi utilizado um equipamento com as seguintes características:

- Processador Core i5 da 3ª geração;
- 8Gb de memória RAM;
- Armazenamento em HDD 500Gb;
- Sistema operacional Windows 10 64bits;

Neste equipamento foi instalado o XAMPP com Apache 5 e PHP 7.1 para a execução dos testes. Com as execuções dos testes foram gerados dois arquivos, o primeiro um arquivo JSON compactado e o outro um arquivo XML compactado. Todos os testes foram executados utilizando a mesma base de dados, a Sakila, da qual foram gerados arquivos JSON e XML de 1 a 1 milhão de registros extraídos do banco de dados e, posteriormente, compactados utilizando o algoritmo Deflate [20].

Para se garantir a consistência dos resultados os testes foram realizados repetidas vezes, 15 para cada teste de compactação e tempo de execução. Os resultados apresentados na Seção 5 variaram de maneira não significativa, em torno de milisegundos, quando medido o tempo de execução e poucos bytes, quano medido o tamanho do arquivo compactado.

Todos os materiais utilizados para a execução dos testes estão disponíveis para download no link que consta em Níckolas D. da Silva et al. [11].

## 5. RESULTADOS

A Tabela 1 ilustra o tamanho dos arquivos XML e JSON em função do número de registros no banco de dados e compara-os em termos de diferença percentual. Como pode ser visto, o formato XML para os mesmos dados e para o mesmo número de itens ocupa mais espaço de armazenamento, porém, de somente 30% a mais do que o formato JSON para arquivos com quantidade de registros superior a 1000, se mantendo essa diferença até 1 milhão de registros.

| TAMANHO (em bytes) | | | |
|---|---|---|---|
| REGISTROS | JSON | XML | XML x JSON |
| 1 | 411 | 564 | 37% |
| 10 | 4.017 | 5.456 | 36% |
| 1.000 | 407.543 | 541.046 | 33% |
| 10.000 | 4.151.543 | 5.410.046 | 30% |
| 20.000 | 8.311.543 | 10.820.046 | 30% |
| 30.000 | 12.471.543 | 16.230.046 | 30% |
| 50.000 | 20.806.078 | 27.050.046 | 30% |
| 100.000 | 41.656.078 | 54.100.046 | 30% |
| 200.000 | 83.356.078 | 108.200.046 | 30% |
| 300.000 | 125.056.078 | 162.300.046 | 30% |
| 500.000 | 208.456.078 | 270.500.046 | 30% |
| 1.000.000 | 417.020.613 | 541.000.046 | 30% |

Tabela 1 – Comparação do tamanho dos arquivos XML e JSON no formato natural.

Na Tabela 2, o tempo levado para compactar os arquivos com 1 e 10 registros é praticamente o mesmo, porém, para os arquivos a partir de 1.000 registros, o tempo de compactação do arquivo XML é até 83% mais rápido que a compactação para o arquivo JSON com a mesma quantidade de registros. A partir de 50 mil essa diferença diminue para 6%, mantendo essa diferença para arquivos com menos de 500 mil, quando os dois arquivos passam a ter o mesmo tempo de compactação (500 mil e 1 milhão de registros).

| TEMPO PARA COMPACTAR (em segundos) | | | |
|---|---|---|---|
| REGISTROS | JSON | XML | XML x JSON |
| 1 | 0,0001 | 0,0001 | 0% |
| 10 | 0,0001 | 0,0001 | 0% |
| 1.000 | 0,0206 | 0,0035 | -83% |
| 10.000 | 0,0535 | 0,0397 | -26% |
| 20.000 | 0,1056 | 0,0801 | -24% |
| 30.000 | 0,1652 | 0,1221 | -26% |
| 50.000 | 0,2175 | 0,2052 | -6% |
| 100.000 | 0,4060 | 0,3797 | -6% |
| 200.000 | 0,8340 | 0,7673 | -8% |
| 300.000 | 1,2190 | 1,2907 | 6% |
| 500.000 | 1,9645 | 1,9607 | 0% |
| 1.000.000 | 3,9219 | 3,9612 | 1% |

Tabela 2 – Comparação do tempo de compactação dos aruivios XML e JSON.

Na análise do tamanho do arquivo após a compactação, ilustrado na Tabela 3, notamos que o arquivo JSON é menor do que o XML apenas para 1 registro, para as demais quantidades de registros, o arquivo XML é menor. Sendo o arquivo com 1.000 registros o menor (- 92%), para arquivos a partir de 100 mil registros, a diferença do tamanho se mantém em torno de 30% menor que os arquivos JSON com mesma quantidade de registros.

| TAMANHO COMPACTADO (em bytes) | | | |
|---|---|---|---|
| REGISTROS | JSON | XML | XML x JSON |
| 1 | 275 | 330 | 20% |
| 10 | 955 | 384 | -60% |
| 1.000 | 39.999 | 3.009 | -92% |
| 10.000 | 77.365 | 26.599 | -66% |
| 20.000 | 114.375 | 52.828 | -54% |
| 30.000 | 151.408 | 79.039 | -48% |
| 50.000 | 225.451 | 131.489 | -42% |
| 100.000 | 410.588 | 262.586 | -36% |
| 200.000 | 780.819 | 524.780 | -33% |
| 300.000 | 1.151.048 | 786.987 | -32% |
| 500.000 | 1.891.532 | 1.311.387 | -31% |
| 1.000.000 | 3.742.744 | 2.622.381 | -30% |

Tabela 3 – Comparação do tamanho dos arquivos XML e JSON compactados.

## 6. CONCLUSÃO

Este artigo apresenta uma comparação dos formatos de representação de dados XML e JSON no que se refere ao tamanho do arquivo compactado e ao tempo de compactação.

A comparação dos dois formatos foi obtida através das funções criadas por Níckolas Daniel [11]. As funções foram então analisadas com base em 3 critérios: tamanho sem compactação, tempo de compactação e tamanho após a compactação. A compactação foi feita usando a ferramenta gzip [12].

Cada um dos 3 critérios foi testado no banco SAKILA [11], aplicando o teste em 1,10,100 até 1 milhão de registros, repetindo 15 vezes cada experimento, observou-se que não houve variação significativa nos resultados, em torno de milisegundos, quando medido o tempo de execução e poucos bytes quando medido o tamanho.

Como uma avaliação geral, observamos que, para todos os registros obtidos, exceto para arquivos com 1 e 10 registros, o formato XML é mais eficiente em termos de tempo de compactação e tamanho compactado. Em alguns casos chegou a ter um tempo de compactação variando de 25% a 83% mais rápido do que o formato JSON. O tamanho compactado chegou a ser 92% menor, aumentando conforme o número de registros cresce. Para 1 milhão de registros o arquivo XML se manteve em 30% menor do que em JSON. O formato XML oferece melhor suporte para alguns aplicativos que exigem envio de estruturas complexas heterogêneas, devendo este estudo contribuir para um melhor entendimento de como transportar tais arquivos.

Uma explicação plausível para esse comportamento é que, em geral, os algoritmos de compactação identificam estruturas repetidas nos arquivos antes de sua compactação e as retiram do arquivo compactado, reconstituindo-as na descompactação. Logo a verbosidade de XML é retirada do arquivo compactado, tornando-o menor que os arquivos JSON campactados.

A possibilidade de comparar com outros formatos representa um desenvolvimento futuro que será investigado em detalhes.

## 7. Referências


[1] T. Anderson, 2004. http://www.itwriting.com/xmlintro.php

[2] Extensible markup language (xml) 1.0 (Fifth edition). W3C, 2008. https://www.w3.org/TR/xml/

[3] U. Hilger, “Article: Client/server with java and xml-rpc,” 2005. http://articles.lightdev.com/csjxml/csjxml_article.pdf.

[4] JSON. json.org. http://www.json.org

[5] S. Klarr, “Javascript: What is json?,” 2007. http://www.scottklarr.com/topic/18/javascriptwhat-isjson

[6] J. F. E. v. d. V. D. A. J. D. A. W. L. M. David Hunter, Jeff Rafter, “Beginning xml,” 4th edition, pp. 6-8, 2007.

[7]W3C Document Object Model. W3C, 2005. http://www.w3.org/DOM

[8] Alonso G., Casati F., Kuno H., Machiraju V. (2004) Web Services. In: **Web Services. Data-Centric Systems and Applications**. Springer, Berlin, Heidelberg

[9] Web service. In: Wikipédia: a enciclopédia livre. Disponível em:< https://pt.wikipedia.org/wiki/Web_service> Acesso em 10 jul 2019

[10] Web APIs. PHPSP. Disponível em https://phpsp.org.br/artigos/web-apis-xml-ou-json/. Acesso 10 jul 2019

[11] Níckolas D. da Silva. Beanstalkd Queue. Dispon´ıvel em: https://github.com/nawarian/benchmark_xml_vs_json. Acesso em: abril de 2019.

[12] GZIP. Disponível em https://www.gzip.org/ Acesso em abril de 2019.

[13] Z.U. Haq, G.F. Khan and T. Hussain, “A Comprehensive analysis of XML and JSON web Technologies”, New Developments in Circuits, Systems, Signal Processing, Communications and Computers, pp. 102-109, 2013

[14] S. Zunke and V. D’Souza, “JSON vx XML: A Comparative Perfomance Anaysis of Data Exchange Formats”, IJCSN International Journal of Computer Science and Network, Volume 3, Isue 4, ISSN 2277-5420, www.IJCSN.org, 2014

[15] K. Ishwarjit, K. Sharanpreet and K. Gurinder, “Accessing Remote Database in IOS Application using JSON Parsing with Objective-C”, International Journal of Advanced Technology in Engineering and Science, Vol. No. 5, No. 1, www.ijates.com, 2017.

[16] Nurseitov, Nurzhan & Paulson, Michael & Reynolds, Randall & Izurieta, Clemente. (2009). Comparison of JSON and XML data interchange formats: A case study. 22nd International Conference on Computer Applications in Industry and Engineering 2009, CAINE 2009. 157-162.

[17] Šandrih, Branislava & Tošić, Dušan & Filipović, Vladimir. (2017). Towards Efficient and Unified XML/JSON Conversion - a New Conversion Method. Transactions on Internet Research (TIR), 13(1):58–64, January 2017. Transactions on Internet Research (TIR). 13. :58–64.

[18] Breje, Anca-Raluca & Gyorodi, Robert & Győrödi, Cornelia & Zmaranda, Doina & Pecherle, George. (2018). Comparative study of data sending methods for XML and JSON models. International Journal of Advanced Computer Science and Applications. 9. 10.14569/IJACSA.2018.091229.

[19] PHP.NET http://www.php.net/. Acesso em dezembro de 2019.

[20] RFC 1951 - DEFLATE Compressed Data Format Specification version http://www.faqs.org/rfcs/rfc1951.html. Acesso em dezembro de 2019.

[21] DEFLATE – Multimidia, http://multimedia.ufp.pt/codecs/compressao-sem-perdas/codificacao-estatistica/codificador-qm/deflate/. Acesso em dezembro de 2019.

[22] PHP: gzdeflate - Manual, https://www.php.net/manual/pt_BR/function.gzdeflate.php. Acesso em dezembro de 2019.