4
TML (Template Model Language)

Ao contrário das demais linguagens, a TML não define a estrutura da informação, como listas ou loops, nem tampouco a fonte, ou a forma de representação.

Novas formas de representação da informação podem surgir. Desta forma, separar a camada de modelo da linguagem permite uma maior flexibilidade. A adição de regras à linguagem a torna mais complexa, porém expressiva. Linguagens de templates já são suficientemente expressivas por possibilitarem a utilização de modelos, logo, a inclusão de regras deve ser fortemente ponderada.

A linguagem de definição de mapeamentos proposta é orientada a templates, i.e., fornece uma estrutura que orienta os usuários no processo de definição de mapeamentos de fontes de dados para RDF, e pode ser utilizada em conjunto com qualquer estrutura de informação como, por exemplo, formatos utilizados pela Web Semântica, e.g. XML, JSON, N3 e N4.

4.1 Visão Geral

TML baseia-se nas estruturas, tipicamente encontradas em documentos, que dividimos em simples e compostas. As estruturas simples são aquelas que geralmente carregam um valor, um literal, já as estruturas compostas são aquelas que encapsulam mais de uma estrutura simples, estruturas compostas ou ambas. Se analisarmos qualquer documento, certamente veremos que toda estrutura pode ser expressa através desses dois tipos de elementos. Esse mesmo conhecimento é utilizado, por exemplo, na definição das marcações de um documento XML. Dessa forma, um vetor contendo uma cadeia de caracteres, pode ser definido através de um elemento composto e vários elementos simples do tipo cadeias de caracteres (*Strings*), como exemplificado na Figura 6.

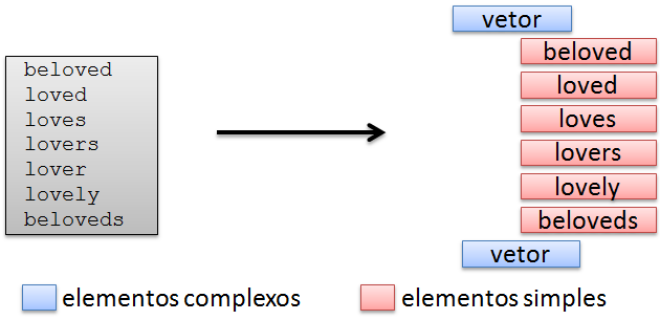

Figura 6. Representação de um vetor em uma estrutura de dados utilizando elementos simples e complexos.

Um relacionamento entre duas tabelas pode ser interpretado como uma composição de dois elementos compostos, onde o primeiro deles representa uma determinada propriedade do segundo, como exemplificado na Figura 7.

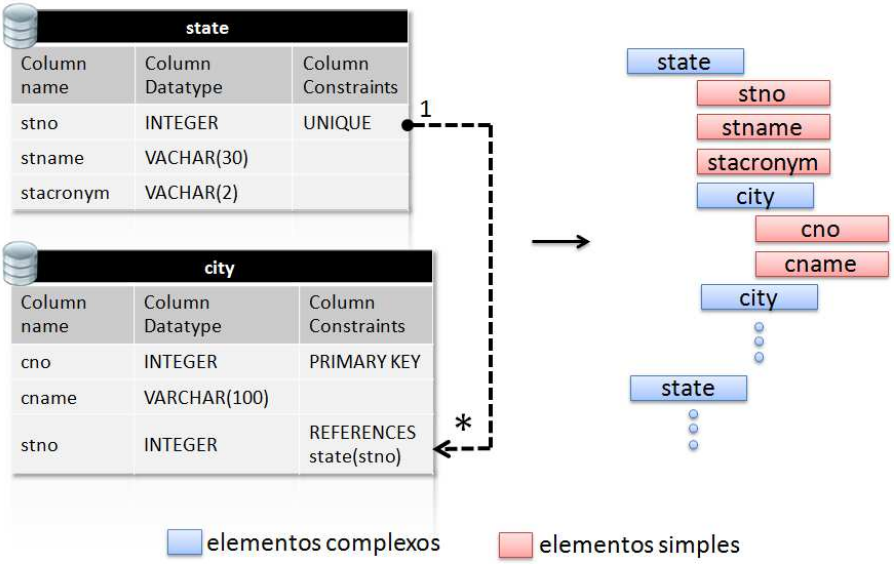

Figura 7. Representação de uma estrutura formada por dois elementos compostos, onde a Tabela cidade (*city*) representa uma propriedade composta da estrutura estado (*state*).

Já que estamos convencidos de que qualquer estrutura pode ser organizada dessa forma, definindo hierarquias, propriedades e tipos de objetos, é razoável utilizar essas hierarquias para definir a estrutura da informação, e não o inverso. Ou seja, o que tentamos com o TML é definir onde uma determinada informação deve ocorrer em certa estrutura e utilizar a estrutura da informação de origem para determinar a quantidade de ocorrências dessa informação.

A utilização do TML permite portar a informação de um contexto de origem para um novo contexto, definido pelo usuário. TML também não especifica o tipo de estrutura de destino, o que permite portar para diversas outras estruturas, embora, no contexto deste trabalho exploramos apenas a conversão para o XML e triplas.

TML trabalha com o mapeamento de coleções de elementos. Assim, como em uma Tabela de banco de dados, os elementos contidos na coleção possuem os mesmos atributos em comum. Uma vez que as coleções estão definidas e que a estrutura de destino é conhecida, o mapeamento deve ser realizado por meio das seguintes regras, exemplificadas na Figura 8:

1. As coleções de objetos deve seguir o padrão, “#{coleção.atributo}” ou “#{coleção}”, onde:
    a. ‘#’ é um indicador (token) definido pela aplicação que determina o início de uma expressão interpretável pela máquina de template;
    b. ‘{‘ e ‘}’ determinam respectivamente, o início e o fim da expressão;
    c. ‘coleção’ representa o nome da coleção;
    d. ‘atrubuto’ representa o atributo que se deseja mapear;
2. Caso um elemento tenha sido mapeado com um atributo da coleção, a estrutura resultante conterá um elemento idêntico ao mapeado, para cada indivíduo da coleção, na mesma ordem e com os mesmos valores.
3. Caso uma estrutura tenha sido mapeada com a mesma coleção em níveis diferentes, a replicação da estrutura se dará no elemento de nível mais alto.
4. A utilização apenas do nome da coleção determina que se deva atribuir o valor vazio à estrutura de destino.

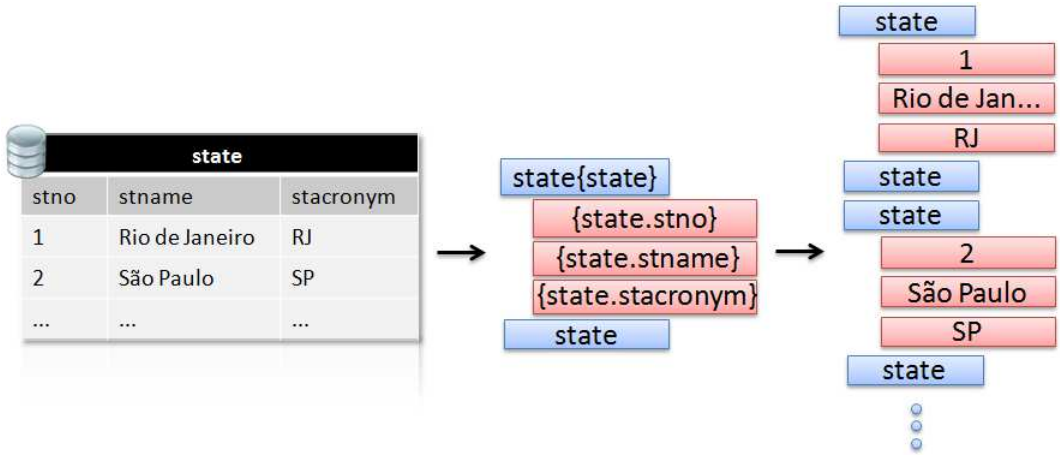

Figura 8. Representação dos dados a partir de um template genérico utilizando TML.

Na Figura 9, abaixo, descrevemos uma base de dados, definimos uma coleção e serializamos a informação extraída da coleção em três estruturas diferentes – JSON, N3 e XML – utilizando chaves como marcação.

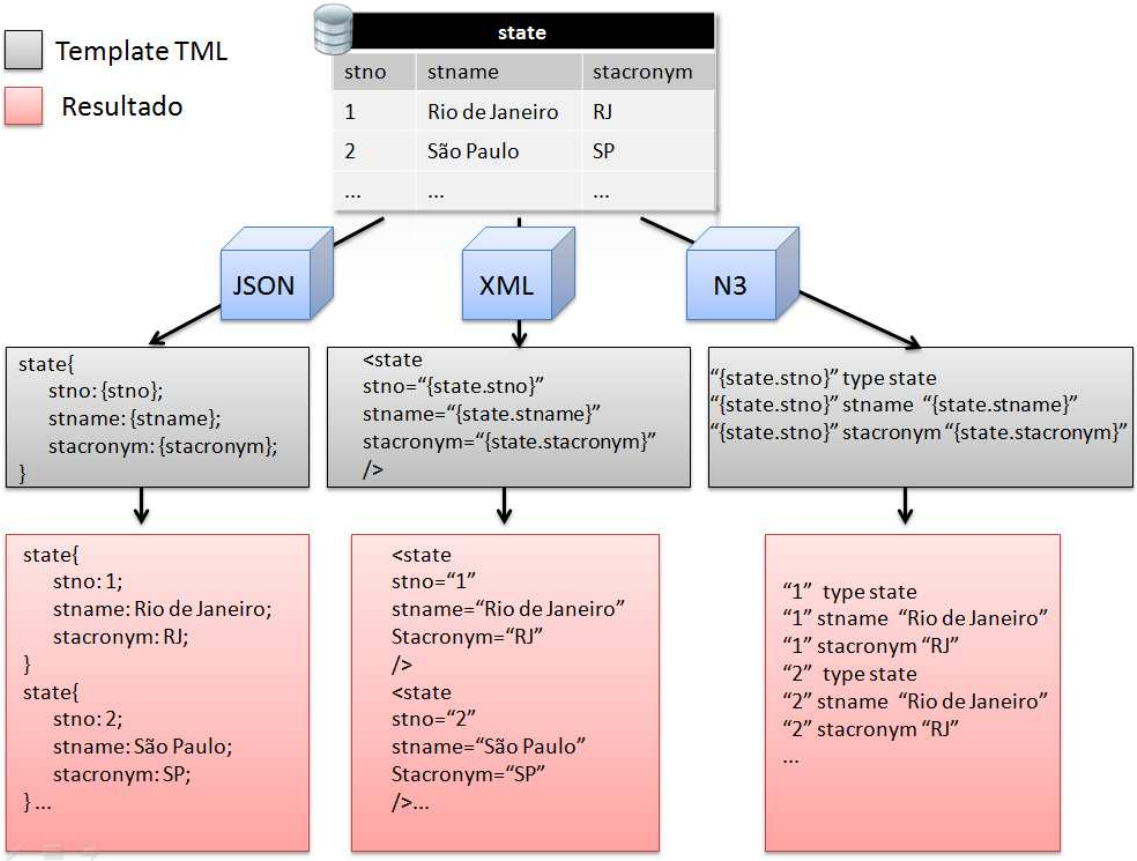

Figura 9. Resultado da conversão dos dados contidos em uma coleção *state* para diferentes formatos de serialização (JSON, XML e N3) utilizando TML.

4.2 URI

Um passo importante na publicação de dados no formato Linked Data é a criação das URIs. As URIs constituem um fator importante na publicação de dados porquê são através delas que os recursos serão recuperados e acessados.

Através da TML, as URIs poderão ser criadas dinamicamente, em tempo de execução. Seguindo as regras já mencionadas da TML, usuários poderão criar templates de URIs que poderão ser preenchidas pelos valores das coleções. No exemplo hipotético, abaixo, a URI possui um campo fixo *http://.myserver.com/* e outro dinâmico *#{state.stacronym}*. Se esta configuração for submetida a um interpretador de TML, o resultado deverá ser uma URI para cada elemento da coleção mapeada formada pelo campo fixo e acrescida do campo dinâmico. É

importante notar, que não existe um limite para o número de coleções mapeadas, coleções poderão ser concatenadas, como no exemplo da Figura 10.

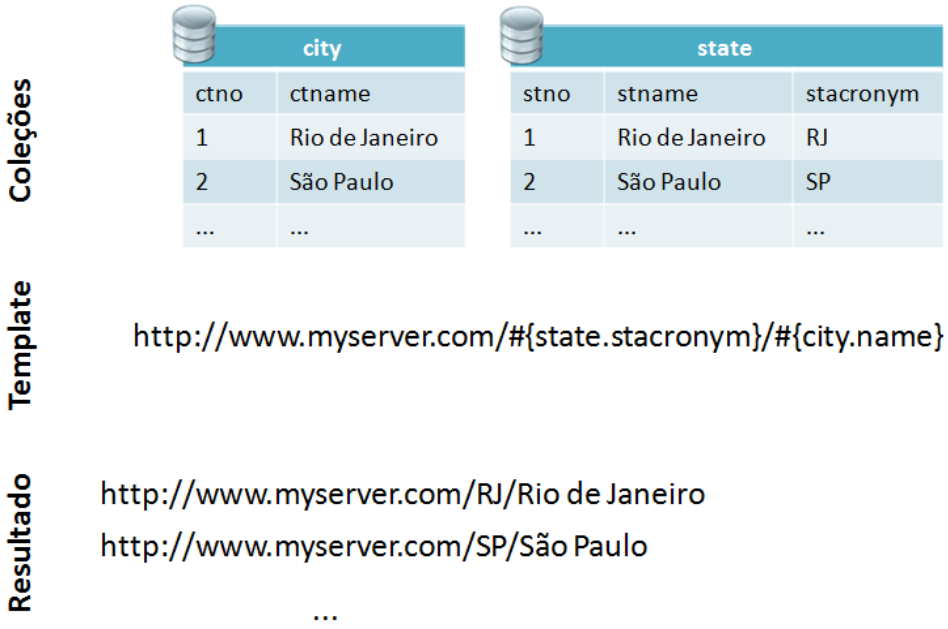

Figura 10. Exemplo de criação de URI dinâmica utilizando TML.

4.3 Limitações

Há obviamente limitações que impedem o uso da aplicação pura de TML. Questões, por exemplo, relativas à ambiguidade e à extração das informações das bases de origens deverão ser posteriormente tratadas pela aplicação.

Como pôde ser observado não é possível definir a fonte de informação de origem, ao invés disso, a linguagem assume que todas as bases estão convertidas para um mesmo formato, o formato de coleções. A partir daí, o mapeamento é realizado de maneira transparente para as estruturas de destino, seguindo as quatro regras da TML apresentadas anteriormente.

É importante ressaltar que fatores relativos à cardinalidade dos elementos das estruturas de origem, ou mesmo o formato, deverão estar de acordo com o esquema alvo, de outra forma, estruturas inválidas poderão ser geradas. Comparativamente TML pode ser vista como uma linguagem para definição da estrutura de destino e suas correspondências às estrutura de origem.

Para exemplificar a questão da ambiguidade, tomemos como exemplo o caso da Figura 6, em que há um mapeamento de duas coleções, estados e cidades. Observe que através da aplicação pura de TML não é possível definir como será o resultado obtido pela aplicação. Note que poderíamos ter dois resultados possíveis, Figura 11: no primeiro, cada estado conteria uma cópia de cada elemento cidade; no segundo, cada estado conteria apenas os elementos cidades pertencentes ao estado, sendo, neste caso, o resultado esperado.

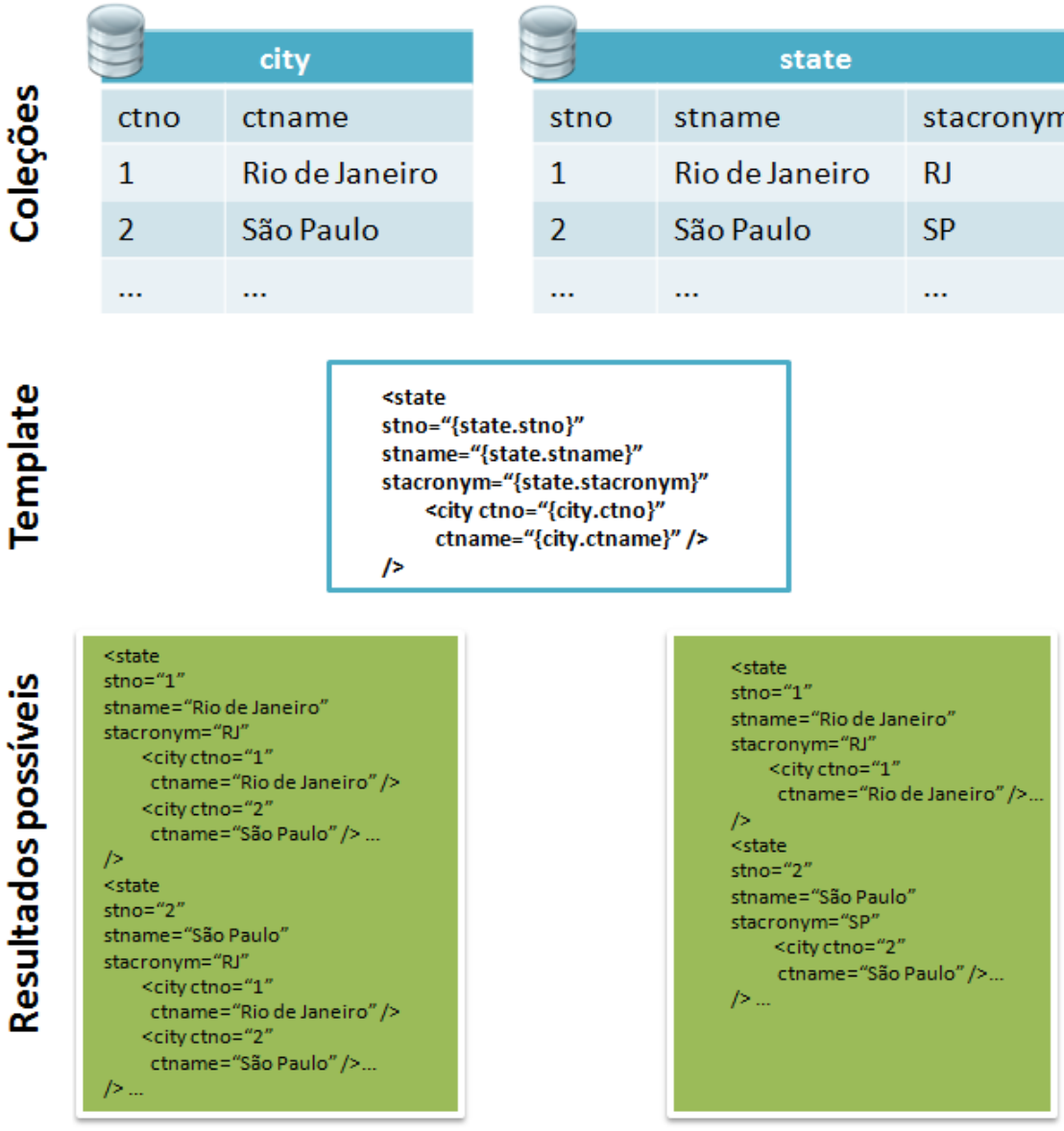

Figura 11. Possíveis resultados de um mapeamento utilizando TML em uma base.

Em resumo, com a TML, não é possível definir assertivas para a estrutura de destino como, por exemplo, cardinalidade, ao invés disso TML deixa esta validação por conta do próprio esquema do vocabulário alvo. Também não é possível explicitar a base de origem nem tão pouco resolver questões relativas à ambiguidade, estas questões devem ser resolvidas pela própria aplicação, através de um método adicional, seja por meio da adição de regras ao TML, ou, seja através de mapeamentos adicionais.

4.4 Utilização de templates em estruturas sensíveis ao contexto

Nosso estudo estaria completo não fosse a existência de estruturas mais complexas que as demonstradas anteriormente, onde os templates apenas são uma reflexão das estruturas de origem. A existência dessas estruturas determinou a utilização do nome da coleção para indicar em qual nível a estrutura será replicada a cada novo elemento inserido.

Vejamos o caso, por exemplo, de um documento HTML, como ilustrado na Figura 11; O documento HTML é formado por estruturas de dados que são sensíveis ao contexto. A alteração de qualquer atributo ou ordem de qualquer elemento influenciará na maneira com que documento será interpretado pelos *browsers*. O mesmo não pode ser afirmado para as estruturas apresentadas anteriormente, onde a ordem de qualquer elemento ou atributo na estrutura, não causaria prejuízo na representação da informação.

Embora a marcação do nível da estrutura que será replicado possa ser feita utilizando um atributo da coleção, deve-se notar que sua utilização implica necessariamente na adição de um determinado valor à estrutura de destino, o que não ocorre utilizando apenas o nome da coleção, como exemplificado na Figura 12.

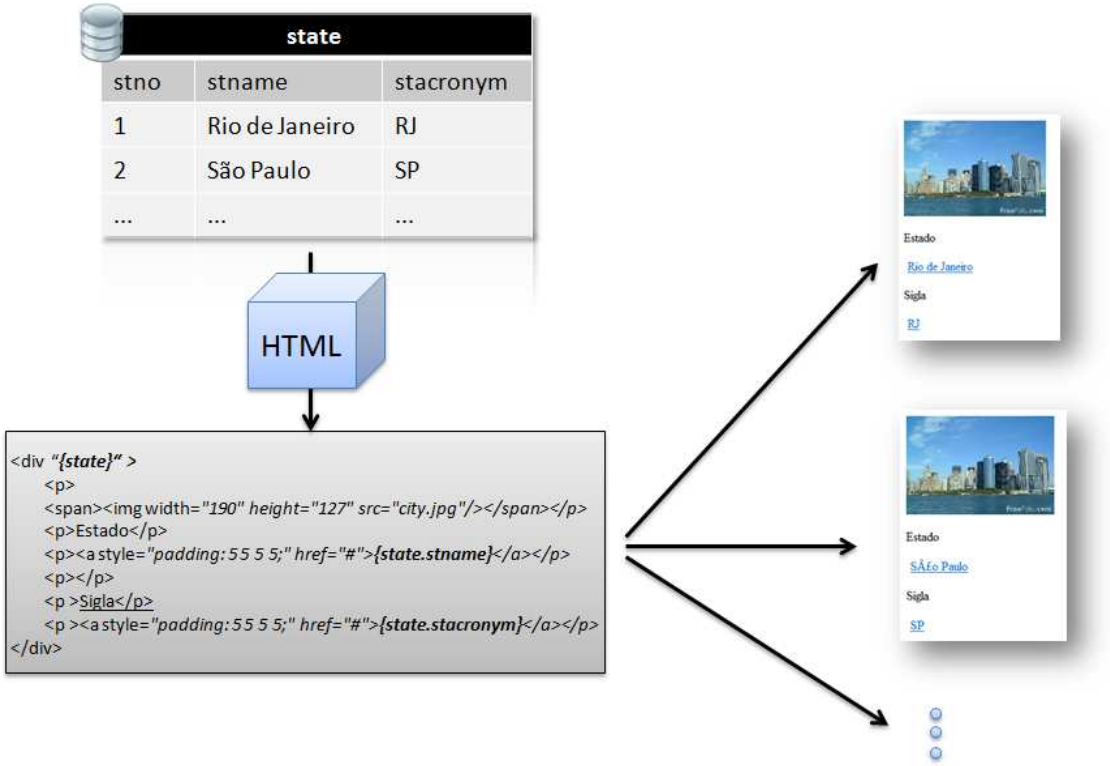

Figura 12. Utilizando templates em estruturas sensíveis ao contexto.

4.5 Escrevendo um compactador de texto simples utilizando TML

Tanto para promover, como para demonstrar a flexibilidade da linguagem proposta, optamos por explorar a construção de um compactador de texto baseado na TML.

Imagine um fragmento de texto extraído de qualquer documento, livro, jornal ou site. Existe um padrão entre todos eles, a utilização de um vocabulário, composto por N palavras, verbos, adjetivos, pronomes, substantivos, etc., além de uma série de N caracteres especiais de acentuação que se encontram ao longo de várias linhas. É possível perceber também que estes M caracteres e N palavras se repetem de uma a K vezes ao longo do texto (onde K é um número maior que zero). Se isso é verdade, podemos afirmar que um texto é um conjunto de linhas formadas por um determinado conjunto de cadeia de caracteres que se repetem ao longo do documento, também podemos afirmar que é possível navegar pelas linhas e identificar as ocorrências dessas cadeias de caracteres.

A partir do momento que podemos identificar as cadeias de caracteres existentes ao longo do texto, para realizar a compactação, basta substituir a cadeia no texto por um marcador que mapeie a cadeia listada ao lugar onde ela aparece no texto. Supondo que o marcador contenha uma cadeia de caracteres menor que a palavra mapeada, é possível afirmar que: seja K o número de cadeias de caracteres substituídas pelo marcador e L a diferença entre o tamanho da palavra pelo tamanho do marcador, teremos uma compactação na ordem de (K)*L. Aplicando essa lógica recursivamente às M cadeias restantes teremos uma ordem total de M*(K)*L, , como exemplificado na Figura 13.

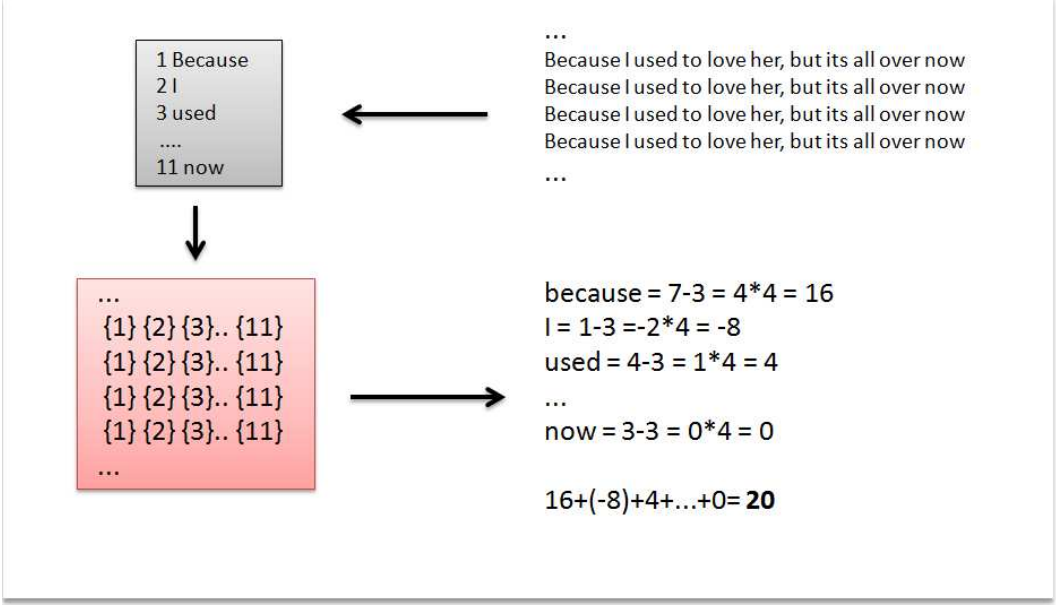

Figura 13. Exemplo de compactação de texto utilizando TML. No exemplo acima utilizamos a linguagem TML para realizar uma compactação de um trecho da música *It´s all over now* do grupo Rolling Stones. Ao final do processo é possível perceber que houve uma compactação de vinte caracteres do texto original.

4.6
Mapeamento direto utilizando TML

O termo mapeamento direto é comumente utilizado para descrever o processo que consiste na criação de instâncias RDF a partir de um esquema retirado de um banco de dados, no qual as Tabelas são mapeadas para classes e as

colunas para propriedades. Embora esse processo seja vastamente debatido no plano de trabalho da W3C chamado "*A Direct Mapping of Relational Data to RDF*" [67], o termo é vastamente empregado em uma variedade de ferramentas, a exemplo dos *Wrappers*.

No mapeamento por templates, o esquema das coleções é utilizado para mapear as informações na estrutura de dados. No mapeamento direto, o esquema do modelo do banco de dados é utilizado na definição das classes e propriedades. Se é possível gerar o esquema automatizado a partir do esquema do modelo do banco de dados, também é possível gerá-lo a partir de coleções. Comparativamente podemos enxergar as coleções como tabelas, uma vez que estas possuem o mesmo conjunto de atributos, e realizar o mapeamento convertendo o esquema da coleção em classes e propriedades. Coleções podem não ter chave primária, no entanto a técnica de mapeamento direto especifica que não havendo chaves primárias, a posição da tupla deva ser tomada como identificador. Ao final, cada coleção será representada por uma classe como demonstrado na Figura 14 abaixo.

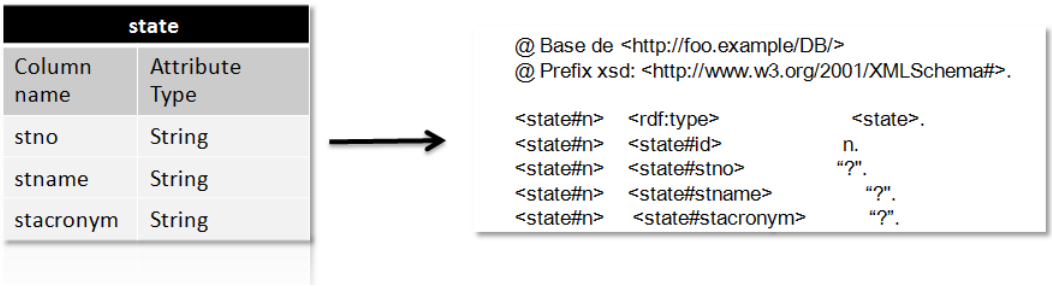

Figura 14. Demonstração do mapeamento da coleção *state* para triplas utilizando a estratégia de mapeamento direto. Neste exemplo onde *n* representa o número da ocorrência da tupla na coleção e o caracter '*?*' a informação contida na tripla do respectivo atributo mapeado.

Uma vez que demonstramos que a o mapeamento direto de coleções é possível, outro passo seria a geração automatizada do template, mas esse processo se torna trivial uma vez que o template também utiliza o esquema das coleções no mapeamento das informações. Dessa forma, para cada coleção existirá uma classe de mesmo nome e atributos conforme a Figura 15.

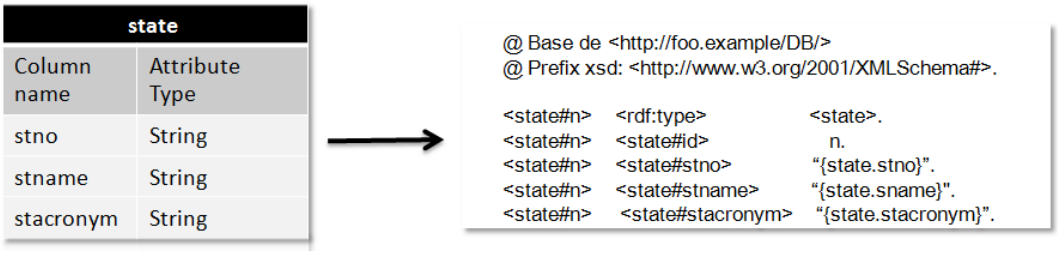

Figura 15. Template de triplas gerado a partir da coleção state, utilizando a técnica de mapeamento direto.