(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2010-267092
(P2010-267092A)
(43)公開日　平成22年11月25日(2010.11.25)

(51)Int.Cl.		F I		テーマコード（参考）
G06F 17/21	(2006.01)	G06F 17/21	501T	5B109

審査請求 未請求　請求項の数 8　ＯＬ　（全 13 頁）

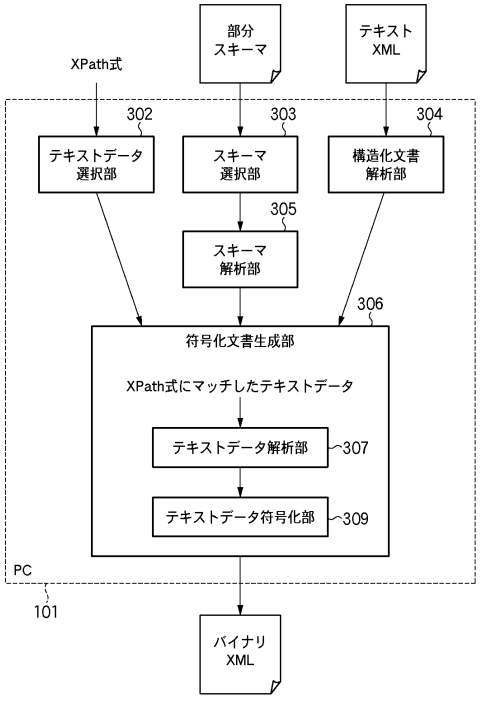
(21)出願番号　特願2009-118047 (P2009-118047)
(22)出願日　平成21年5月14日 (2009.5.14)

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100076428
弁理士　大塚　康徳
(74)代理人　100112508
弁理士　高柳　司郎
(74)代理人　100115071
弁理士　大塚　康弘
(74)代理人　100116894
弁理士　木村　秀二
(74)代理人　100130409
弁理士　下山　治
(74)代理人　100134175
弁理士　永川　行光

最終頁に続く

(54)【発明の名称】情報処理装置、情報処理方法

(57)【要約】　（修正有）
【課題】サイズの大きなテキストデータを属性値又は要素内容として含むテキストＸＭＬからバイナリXMLへの変換効率を改善するための技術を提供する。
【解決手段】テキスト形式の構造化文書を取得する手段と、前記構造化文書を解析し、テキストデータを検出する手段と、前記テキストデータを子要素に変換することで前記構造化文書を変換し、変換した構造化文書に基づいてバイナリ形式の構造化文書を生成する生成手段とを備えることを特徴とする情報処理装置。
【選択図】図４



【特許請求の範囲】
【請求項１】
テキスト形式の構造化文書を取得する手段と、
前記構造化文書を解析し、テキストデータを検出する手段と、
前記テキストデータを子要素に変換することで前記構造化文書を変換し、変換した構造化文書に基づいてバイナリ形式の構造化文書を生成する生成手段と
を備えることを特徴とする情報処理装置。
【請求項２】
前記テキストデータは、前記構造化文書に含まれる属性値であることを特徴とする請求項１に記載の情報処理装置。
【請求項３】
前記テキストデータは、前記構造化文書に含まれる要素内容であることを特徴とする請求項１に記載の情報処理装置。
【請求項４】
前記生成手段は、
前記テキストデータの構造を定義するスキーマを取得する手段と、
前記スキーマを用いて前記テキストデータを子要素に変換することで前記構造化文書を変換する変換手段と
を備えることを特徴とする請求項１乃至３の何れか１項に記載の情報処理装置。
【請求項５】
前記変換手段は、
前記テキストデータにおいて、前記スキーマのgroup要素以下で定義された列挙値に該当する変数については、当該列挙値をタグ名とする開始タグに変換する手段と、
前記テキストデータにおいて、前記変数に後続する値については、前記開始タグに後続するデータとする手段と
を備えることを特徴とする請求項４に記載の情報処理装置。
【請求項６】
テキスト形式の構造化文書を取得する工程と、
前記構造化文書を解析し、テキストデータを検出する工程と、
前記テキストデータを子要素に変換することで前記構造化文書を変換し、変換した構造化文書に基づいてバイナリ形式の構造化文書を生成する生成工程と
を備えることを特徴とする情報処理方法。
【請求項７】
コンピュータを請求項１乃至５の何れか１項に記載の各手段として機能させるためのコンピュータプログラム。
【請求項８】
請求項７に記載のコンピュータプログラムを格納した、コンピュータ読み取り可能な記憶媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、構造化文書を処理する技術に関するものである。
【背景技術】
【０００２】
これまで、XMLデータのフォーマットは、一般的にテキスト形式であった。しかし、XMLデータは、データ構造を表現するために冗長なデータを必要とし、コンピュータが読み書きするのに時間がかかるという問題があるため、近年では、バイナリXML技術が注目されている。
【０００３】
例えば、W3Cの標準仕様としてEfficient XML Interchange (htt:／／www.w3.org／XML

／EXI／)がある。また、ISOの標準仕様としては、Fast Infoset(http:／／www.iso.org／iso／en／CatalogueDetailPage.CatalogueDetail?CSNUMBER=41327&scopelist=PROGRAMME)がある。

【０００４】

これらの技術は、XMLデータに含まれる要素名や属性名などの各ボキャブラリを、XMLデータ内での出現順に番号を振って符号化することによって、データサイズを小さくすることが可能であった。符号とボキャブラリとの対応を示す表は、符号化テーブルと呼ばれる。バイナリXML形式のデータは、従来のテキスト形式のXMLデータに比べて、高速に読み書きすることが可能であった。

【０００５】

また、属性値や要素内容を従来のような文字列型のデータではなく、整数型、浮動小数点型や、独自形式の圧縮アルゴリズムを用いてエンコードすることにより、データサイズを更に小さくすることが可能であった。また、それらデータを解析する際には、従来のようなアプリケーションによる文字列データから数値データへの変換処理の必要がなくなるので、解析にかかる処理時間を更に短くすることができた。

【０００６】

また、XMLデータの文書構造と、含まれる属性値や要素内容の値のデータ型を、スキーマを用いることによって定義することができた。一般的には、それらスキーマは、XMLデータの文書構造と、値のデータ型が厳密にスキーマの定義内容と合っているかどうかを検証するために使用される。W3Cのスキーマの標準仕様としては、XML Schema(http:／／www.w3.org／XML／Schema)がある。また、ISOのスキーマの標準仕様としては、RELAX NG(http:／／www.oasis-open.org／committees／relax-ng／)がある。

【０００７】

また、スキーマを利用した他の技術としては、データバインディングがある。これは、開発時に、スキーマからXMLデータを格納するクラスファイルを自動生成しておき、実行時には、アプリケーションがそのクラスを利用することによって、アプリケーションの開発コストを小さくするものである。スキーマから生成したクラスにXMLデータをバインディングする際には、スキーマで定義した各データ型に基づいて、含まれるテキストデータを変換し、各メンバに格納する。よって、従来のテキストベースのXMLパーサとは異なり、アプリケーションはスキーマから生成したクラスを用いることによって、XMLデータに含まれるテキストデータを各データ型の値として扱うことが可能になる。一般的にはSun MicorsystemsのJAXB(https:／／jaxb.dev.java.net／)や、Relaxer(http:／／www.relaxer.jp／)が利用されている。

【０００８】

また、KDDIのXEUSは、属性値や要素内容において、カンマなどで区切られた数値の配列に対しては、各数値をそれぞれ符号化することによって、XMLデータを効率的に圧縮することが可能であった。

【０００９】

また、スキーマ情報を用いることによって、文書構造の異なる複数のXMLデータ間の変換用スタイルシートを自動生成するものがあった（特許文献１）。また、アプリケーションからXMLデータへのアクセス効率を改善するために、XMLデータに含まれる階層構造が浅くなるように、タグ名を変換するものがあった（特許文献２）。

【先行技術文献】

【特許文献】

【００１０】

【特許文献１】特開２００５－３５２９４５号公報

【特許文献２】特開２００２－２９７５６９号公報

【発明の概要】

【発明が解決しようとする課題】

【００１１】

従来のバイナリXMLでは、属性値や要素内容などに大きなサイズのテキストデータが含まれている場合、バイナリXML形式に符号化してもそれらの圧縮効率が良くないので、データサイズを小さくすることが難しかった。

【００１２】

ここで、W3C標準のSVGを用いて図形オブジェクト（図１（ｂ））を記述したテキスト形式のXMLデータ（図１（ａ））について説明する。このテキスト形式のXMLデータをバイナリXML形式に符号化しても、path要素のd属性の値はそのままテキストデータとして符号化され、バイナリXMLデータ内の文字列テーブルに格納される。よって、XMLデータ内に繰り返し出現する可能性の低い、サイズの大きなテキストデータは、従来のバイナリXML技術を用いても効率的に圧縮することが難しかった。特にSVGにおいては、path要素のｄ属性の値に長いテキストデータを用いて図形オブジェクトを記述する場合が多いので、バイナリXMLによる圧縮効果を得ることが難しかった。

【００１３】

本発明は以上の問題に鑑みてなされたものであり、サイズの大きなテキストデータを属性値又は要素内容として含むテキストＸＭＬからバイナリXMLへの変換効率を改善するための技術を提供することを目的とする。

【課題を解決するための手段】

【００１４】

本発明の目的を達成するために、例えば、本発明の情報処理装置は以下の構成を備える。即ち、テキスト形式の構造化文書を取得する手段と、前記構造化文書を解析し、テキストデータを検出する手段と、前記テキストデータを子要素に変換することで前記構造化文書を変換し、変換した構造化文書に基づいてバイナリ形式の構造化文書を生成する生成手段とを備えることを特徴とする。

【発明の効果】

【００１５】

本発明の構成により、サイズの大きなテキストデータを属性値又は要素内容として含むテキストＸＭＬからバイナリXMLへの変換効率を改善することができる。

【図面の簡単な説明】

【００１６】

【図１】テキスト形式のＸＭＬデータとその描画結果を示す図。
【図２】実施形態に係るシステムの構成例を示す図。
【図３】ＰＣ１０１のハードウェア構成例を示すブロック図。
【図４】ＰＣ１０１の処理機能ブロック図。
【図５】ＰＣ１０１が行う処理のフローチャート。
【図６】ステップＳ１１３の詳細を示すフローチャート。
【図７】部分スキーマの一例を示す図。
【図８】（ａ）はXPath式の例を示す図、（ｂ）はバイナリ形式のＸＭＬデータを示す図。
【図９】各イベントの符号を示す図。
【図１０】名前テーブルを示す図。
【図１１】（ａ）は値テーブルを示す図、（ｂ）は符号のテーブルを示す図。

【発明を実施するための形態】

【００１７】

以下、添付図面を参照し、本発明の好適な実施形態について説明する。なお、以下説明する実施形態は、本発明を具体的に実施した場合の一例を示すもので、特許請求の範囲に記載の構成の具体的な実施例の１つである。

【００１８】

本実施形態では、比較的大きなサイズのテキストデータを属性値として有する属性を含む要素が記された構造化文書としてのテキストXMLを、圧縮効率良くバイナリXMLに変換する為の技術について説明する。

【００１９】
先ず、本実施形態に係るシステムについて、図２を用いて説明する。図２に示す如く、本実施形態に係るシステムは、本実施形態に係る情報処理装置としてのＰＣ（パーソナルコンピュータ）１０１、ネットワークの一例であるＬＡＮ１０２、デジタルカメラ１０３、複合機１０４、ファイルサーバ１０５により構成されている。
【００２０】
デジタルカメラ１０３は、ＰＣ１０１で生成したＸＭＬデータと共に用いる画像を撮像により取得するためのもので、デジタルカメラ１０３により撮像された画像のデータは、ＬＡＮ１０２を介してＰＣ１０１やファイルサーバ１０５に転送される。もちろん、デジタルカメラ１０３の使用方法はこれに限定するものではない。
【００２１】
複合機１０４は、ＰＣ１０１で生成したＸＭＬデータと共に用いる画像を複写により取得するためのもので、複合機１０４により複写された画像のデータは、ＬＡＮ１０２を介してＰＣ１０１やファイルサーバ１０５に転送される。もちろん、複合機１０４の使用方法はこれに限定するものではない。
【００２２】
ファイルサーバ１０５は、デジタルカメラ１０３や複合機１０４から転送された画像のデータを保持すると共に、ＰＣ１０１から転送されたＸＭＬデータも保持する。ファイルサーバ１０５が保持する画像のデータやＸＭＬデータに対しては、ＬＡＮ１０２を介して適宜アクセスすることができる。
【００２３】
ＰＣ１０１は、本実施形態に係る情報処理装置として用いるもので、テキスト形式のＸＭＬデータをバイナリ形式のＸＭＬデータに変換する後述の処理を実行する。次に、図３を用いて、ＰＣ１０１のハードウェア構成例について説明する。
【００２４】
ＣＰＵ２０１は、ＲＯＭ２０２やＲＡＭ２０３に格納されているコンピュータプログラムやデータを用いてＰＣ１０１全体の制御を行うと共に、ＰＣ１０１が行うものとして後述する各処理を実行する。
【００２５】
ＲＯＭ２０２は、コンピュータ読み取り可能な記憶媒体の一例であり、ＰＣ１０１の設定データやブートプログラムなどが格納されている。ＲＡＭ２０３は、コンピュータ読み取り可能な記憶媒体の一例であり、記憶部２０４からロードされたコンピュータプログラムやデータ、ＬＡＮ　Ｉ／Ｆ２０７、ＵＳＢ　Ｉ／Ｆ２０９を介して外部から受信したデータ等を一時的に記憶する為のエリアを有する。また、ＲＡＭ２０３は、ＣＰＵ２０１が各種の処理を実行する際に用いるワークエリアも有する。即ち、ＲＡＭ２０３は、各種のエリアを適宜提供することができる。
【００２６】
記憶部２０４は、コンピュータ読み取り可能な記憶媒体の一例であり、ハードディスクドライブ装置に代表される、大容量情報記憶装置である。記憶部２０４には、ＯＳ（オペレーティングシステム）や、ＰＣ１０１が行うものとして後述する各処理をＣＰＵ２０１に実行させるためのコンピュータプログラムやデータが保存されている。このコンピュータプログラムやデータは、ＣＰＵ２０１による制御に従って適宜ＲＡＭ２０３にロードされ、ＣＰＵ２０１による処理対象となる。
【００２７】
操作部２０５は、キーボードやマウスなどにより構成されており、ＰＣ１０１の操作者が操作することで、各種の指示をＣＰＵ２０１に対して入力することができる。表示部２０６は、ＣＲＴや液晶画面などにより構成されており、ＣＰＵ２０１による処理結果を画像や文字などでもって表示することができる。
【００２８】
ＬＡＮ　Ｉ／Ｆ２０７は、ＰＣ１０１をＬＡＮ１０２に接続する為のもので、ＰＣ１０

１はこのＬＡＮ　Ｉ／Ｆ２０７を介してＬＡＮ１０２に接続されている機器との通信を行う。ＵＳＢ　Ｉ／Ｆ２０９は、ＰＣ１０１をＵＳＢ回線２１０に接続する為のもので、ＰＣ１０１はこのＵＳＢ　Ｉ／Ｆ２０９を介してＵＳＢ回線２１０に接続されている機器との通信を行う。上記各部は何れも共通のバスに接続されており、互いに通信を行うことができる。

【００２９】

なお、以上で説明したシステムの構成やＰＣ１０１の構成は一例であって、後述する処理を実現することができ、且つ後述する処理によって得られるバイナリ形式のＸＭＬデータを用いた応用が可能であれば、他の構成を適用しても良い。

【００３０】

次に、テキスト形式のＸＭＬデータからバイナリ形式のＸＭＬデータへの効率の良い圧縮符号化処理について、図５を用いて説明する。なお、図５に示したフローチャートに従った処理をＣＰＵ２０１に実行させるためのコンピュータプログラムやデータは記憶部２０４に保存されている。係るコンピュータプログラムやデータは、ＣＰＵ２０１による制御に従って適宜ＲＡＭ２０３にロードされ、ＣＰＵ２０１による処理対象となる。ＣＰＵ２０１がこのロードされたコンピュータプログラムやデータを用いて処理を実行することで、ＰＣ１０１は、以下に説明する各処理（図５に示した各ステップにおける処理）を実行することになる。

【００３１】

また、以下では、説明を簡単にするために、圧縮符号化対象のテキスト形式のＸＭＬデータは、図１に示したＸＭＬデータであるものとする。もちろん、以下に説明する処理の本質は、他のテキスト形式のＸＭＬデータであっても同じである。

【００３２】

先ずステップＳ１０２では、ＣＰＵ２０１は、バイナリ形式のＸＭＬデータへの圧縮符号化対象となるテキスト形式のＸＭＬデータを選択する。係る選択は、ＰＣ１０１上で動作しているアプリケーションによって行っても良いし、操作部２０５を介して操作者が入力した選択指示に基づいて行っても良い。上述の通り、ここでは、図１に示したＸＭＬデータが選択されたものとする。係るＸＭＬデータには、効率的に圧縮することの難しいテキストデータ（Ｘｐａｔｈ式にマッチするテキストデータ）を属性値として有する属性（図１ではｄ属性）を含む要素（図１ではｐａｔｈ要素）が含まれている。

【００３３】

次にステップＳ１０３では、効率的に圧縮することの難しいテキストデータに関するデータ型を記述した部分スキーマと、そのデータ型の情報のルートとなる定義名と、を選択し、この部分スキーマを解析し、ＲＡＭ２０３内にＤＯＭツリーを作成する。

【００３４】

なお、部分スキーマと、そのデータ型の情報のルートとなる定義名は、表示部２０６にこれらの選択を行うためのユーザインタフェース画面を表示し、ユーザがこの画面を見ながら操作部２０５を用いて選択指示を入力するようにしても良い。また、部分スキーマと、そのデータ型の情報のルートとなる定義名を、あらかじめアプリケーションロジックに組み込むことで、これらを選択するようにしても良い。

【００３５】

上記の通り、本実施形態では、効率的に圧縮することの難しいデータは、図１におけるｐａｔｈ要素のｄ属性の属性値である。また、RELAX NGを用いて記述した、図１におけるｐａｔｈ要素のｄ属性のデータ型を定義した部分スキーマの一例について、図７を用いて説明する。

【００３６】

図７に例示した部分スキーマでは、define要素を用いて複数のデータ型を記述しているが、ｄ属性の値のデータ型のルートとなる定義名は、”SVG.PathData.datatype”である。然るに、図７の場合、ステップＳ１０３で選択する定義名は、”SVG.PathData.datatype”となる。

【００３７】
図５に戻って、次に、ステップＳ１０４では、RELAX NG仕様書の第４章単純化に基づいて、選択した部分スキーマの文書構造を単純化する。そして、ステップＳ１０５では、XPath式を用いることにより、定義した部分スキーマに対応するテキストデータを選択する。テキスト形式のＸＭＬデータに出現する全てのpath要素のd属性を表現したXPath式の例は、図８（ａ）に示すようになっている。なお、XPath式を用いたテキストデータの選択については、表示部２０６に選択を行うためのユーザインタフェース画面を表示し、ユーザがこの画面を見ながら操作部２０５を用いて選択指示を入力するようにしても良い。また、XPath式をあらかじめアプリケーションロジックに組み込むことで、係る選択を行うようにしても良い。
【００３８】
次に、ステップＳ１０６では、テキスト形式のＸＭＬデータを解析する。係る解析は、テキスト形式のＸＭＬデータの先頭から順次行うものとする。次に、ステップＳ１０７では、テキスト形式のＸＭＬデータからバイナリ形式のＸＭＬデータへの符号化が完了したか否かを判断する。係る判断の結果、完了している場合には本処理は終了し、完了していない場合には処理はステップＳ１０８に進む。
【００３９】
ステップＳ１０８では、テキスト形式のＸＭＬデータ中に、XPath式にマッチするテキストデータが含まれているか否かを判断する。係る判断の結果、含まれている場合（テキスト形式のＸＭＬデータからXPath式にマッチするテキストデータを検出可能である場合）には処理をステップＳ１１０に進め、含まれていない場合には処理をステップＳ１０９に進める。ステップＳ１０９では、テキスト形式のＸＭＬデータをそのまま従来通りにバイナリ形式のＸＭＬデータに符号化する。
【００４０】
ステップＳ１１０では、テキスト形式のＸＭＬデータにおいて、XPath式にマッチしないデータを符号化する。図１のテキスト形式のＸＭＬデータの場合、最初の開始タグsvgはXPath式にマッチするテキストデータを含んでいないので、そのまま開始タグとして符号化する。そして、次のpath要素は、XPath式にマッチするｄ属性を含んでいるので、開始タグpathと、fill、stroke、stroke-width属性を符号化する。なお、XPath式にマッチしたｄ属性とその属性値は、符号化しないでＲＡＭ２０３に保存しておく。
【００４１】
次に、ステップＳ１１１では、XPath式にマッチしたテキストデータの所属先が、属性値か否かを判断する。係る判断の結果、XPath式にマッチしたテキストデータの所属先が属性値である場合には、処理をステップＳ１１２に進め、XPath式にマッチしたテキストデータの所属先が属性値ではない場合には、処理をステップＳ１１５に進める。
【００４２】
ステップＳ１１２では、XPath式にマッチしたテキストデータの所属先としての属性値の属性名をタグ名とする開始タグを符号化する。図１のテキスト形式のＸＭＬデータの例では、XPath式にマッチしたテキストデータは属性値に所属し、その属性名が”ｄ”であるので、”ｄ”という名前を持つ開始タグ＜ｄ＞を符号化する。
【００４３】
次に、ステップＳ１１３では、XPath式にマッチしたテキストデータを先頭から逐次解析し、選択した部分スキーマのデータ型の情報を用いることによって、そのテキストデータを子要素に変換する。
【００４４】
ここで、ステップＳ１１３における処理の詳細について、図６を用いて説明する。まずステップＳ２０２では、部分スキーマのDOMツリー内の内部状態を初期化する。次に、ステップＳ２０３では、テキストデータの解析途中の状態を記憶しておくために、部分スキーマ内の任意のノードを指し示すポインタｐを、ＮＵＬＬや最初のノードへのポインタ値等に初期化する。

【００４５】
次に、ステップＳ２０４では、XPath式にマッチしたテキストデータについての解析が完了したか否かを判断する。係る判断の結果、完了した場合にはステップＳ１１３の処理は終了し、ステップＳ１１４に処理を進める。一方、係る判断の結果、完了していない場合には、処理をステップＳ２０５に進める。なお、テキストデータの解析処理は、テキストデータに含まれる空白文字を区切りの識別子とし、この識別子でテキストデータを分割し、それぞれの分割テキストデータ毎に解析する。
【００４６】
ステップＳ２０５では、分割テキストデータが、部分スキーマのgroup要素以下で定義された列挙値に該当する変数であるか否かを判断する。係る判断の結果、該当する場合には、処理をステップＳ２０６に進め、該当しない場合には、処理をステップＳ２０７に進める。
【００４７】
ステップＳ２０６では、部分スキーマのgroup要素以下で定義された列挙値に該当する変数としての分割テキストデータを、その列挙値をタグ名とする開始タグに符号化する。図７に示した部分スキーマを用いる場合、列挙値は、”Ｍ”、”ｍ”、”Ｌ”、”ｌ”であるので、最初に解析する分割テキストデータ”Ｍ”は、”moveType”を名前として持つdefine要素以下に対応する。然るにこのような場合には、最初に解析する分割テキストデータ”Ｍ”を、”Ｍ”をタグ名とする開始タグに符号化する。そしてその後、ステップＳ２０８では、解析途中の状態を記憶しておくために、例えば図７の場合には、部分スキーマ内の名前が”moveType”のdefine要素以下のchoice要素を指し示すように、ポインタｐを更新する。
【００４８】
一方、ステップＳ２０７では、ポインタｐが指し示す部分スキーマ内の参照先のノードを解析する。そして、分割テキストデータを、そのテキストデータのデータ型に基づいて、１つのテキストイベントとして符号化する。図７に示した部分スキーマを用いる場合、最初の分割テキストデータ”Ｍ”を開始タグとして符号化した後、”Ｍ”に後続する分割テキストデータ”１００”は、部分スキーマで定義されているfloat型で１つのテキストイベントとして符号化する。
【００４９】
そしてその後ステップＳ２０８では、現在ポインタｐが指し示していたノードの兄弟ノードを参照するように、ポインタｐを更新する。従って、分割テキストデータ”１００”を符号化した後、ポインタｐは、”moveType”を名前に持つdefine要素以下の＜data type=”float”＞ノードを参照することになる。このようにして、１つのテキストデータを構成する全ての分割テキストデータについて順次、符号化する。
【００５０】
図５に戻って、次に、ステップＳ１１４では、ステップＳ１１２で符号化した開始タグに対応する終了タグを符号化する。本実施形態では、”ｄ”というタグ名を有する終了タグ＜／d＞を符号化する。一方、ステップＳ１１５では、ステップＳ１１２，Ｓ１１４で符号化した開始タグと終了タグを出力しないで、ステップＳ１１３と同様の処理を行う。
【００５１】
以上説明した符号化処理を図１に示したテキスト形式のＸＭＬデータに対して行うことで、図８（ｂ）に示すバイナリ形式のＸＭＬデータを作成することができる。図８（ｂ）に示したバイナリは、それぞれの単位に対応するバイナリである。
【００５２】
図１のテキスト形式のＸＭＬデータのpath要素に含まれるd属性のデータは、path要素の子要素として含まれるように変換される。また、d属性の値のテキストデータは、”Ｍ”や”Ｌ”といったコマンドをグループ（テキストデータの構造単位）の単位として子要素として変換される。その要素内容には、”Ｍ”や”Ｌ”のコマンドに対して指定する座標情報が、１つ１つのテキストイベントとして格納される。アプリケーションが、”Ｍ”や”Ｌ

”のコマンドに対して指定した座標情報をデコードした際には、各座標情報を別々のテキストイベントとして取得することができる。

【００５３】

図８（ｂ）に示したバイナリ形式のＸＭＬデータに関する各イベントの符号を図９に示す。アプリケーションは事前に各イベントの符号を知っているものとする。また、図８（ｂ）で参照される名前テーブルを図１０に示す。この名前テーブルは、要素や属性の名前と、各名前に割り当てる符号と、各名前のデータサイズを格納したテーブルである。

【００５４】

また、図８（ｂ）に示したバイナリ形式のＸＭＬデータで参照される値テーブルを図１１（ａ）に示す。この値テーブルは、要素内容や属性値の値と、各値に割り当てる符号と、各値のデータ型、各値のデータサイズを格納したテーブルである。図１０，１１（ａ）に示したテーブルは、変換元となるテキスト形式のＸＭＬデータに依存するものであるので、テキスト形式のＸＭＬデータの内容が変われば、図１０，１１（ａ）に示したテーブルの内容もそれに依存して変わる。図１１（ｂ）は、図１１（ａ）に示したテーブルの各値のデータ型として参照する符号のテーブルであり、アプリケーションは事前に知っているものとする。

【００５５】

次に、ＰＣ１０１の処理機能ブロックについて、図４を用いて説明する。構造化文書解析部３０４は、バイナリ形式のＸＭＬデータへの圧縮符号化対象となるテキスト形式のＸＭＬデータを取得し、解析する。

【００５６】

スキーマ選択部３０３は、効率的に圧縮することの難しいテキストデータに関するデータ型を記述した部分スキーマと、そのデータ型の情報のルートとなる定義名と、を選択する。

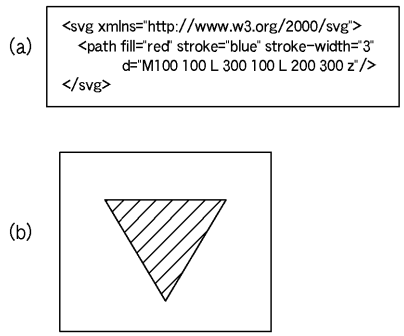
【００５７】

スキーマ解析部３０５は、この部分スキーマを解析し、ＲＡＭ２０３内にＤＯＭツリーを作成する。そして更に、スキーマ解析部３０５は、RELAX NG仕様書の第４章単純化に基づいて、選択した部分スキーマの文書構造を単純化する。

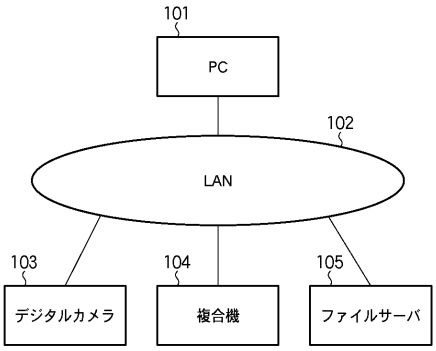
【００５８】

テキストデータ選択部３０２は、XPath式を取得する。符号化文書生成部３０６は、XPath式を用いることにより、定義した部分スキーマに対応するテキストデータを選択する。符号化文書生成部３０６は、テキストデータ解析部３０７、テキストデータ符号化部３０９を有しており、各部は以下のように動作する。テキストデータ解析部３０７は、XPath式にマッチしたテキストデータの解析を行い、テキストデータ符号化部３０９は、上記各符号化処理を行う。

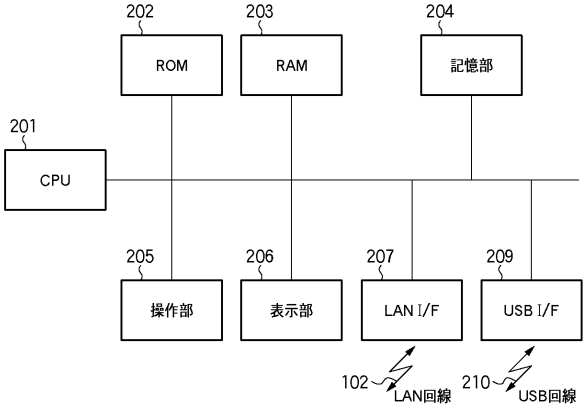
【図１】


<svg xmlns="http://www.w3.org/2000/svg">
<path fill="red" stroke="blue" stroke-width="3"
d="M100 100 L 300 100 L 200 300 z"/>
</svg>
(a)
(b)

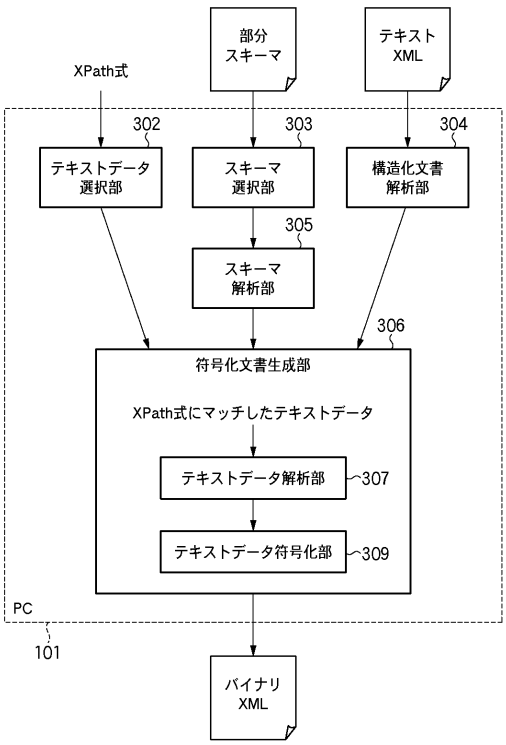
【図２】


101
PC
102
LAN
103
104
105
デジタルカメラ
複合機
ファイルサーバ

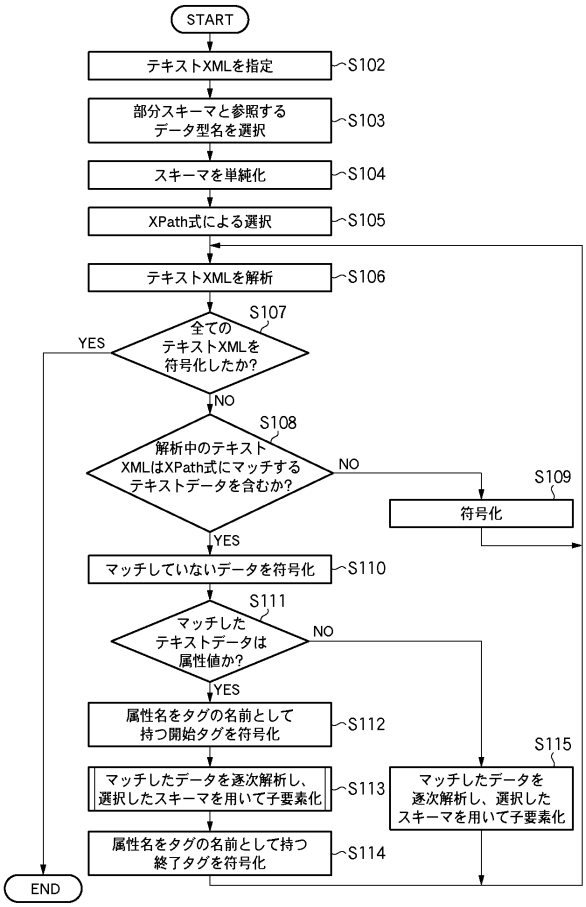
【図３】


202
203
204
ROM
RAM
記憶部
201
CPU
205
206
207
209
操作部
表示部
LAN I/F
USB I/F
102
LAN回線
210
USB回線

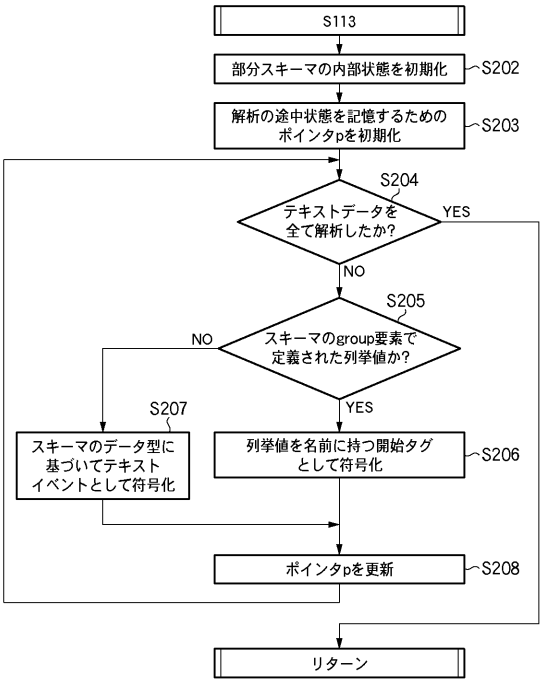
【図４】


部分
スキーマ
テキスト
XML
XPath式
302
303
304
テキストデータ
選択部
スキーマ
選択部
構造化文書
解析部
305
スキーマ
解析部
306
符号化文書生成部
XPath式にマッチしたテキストデータ
テキストデータ解析部
307
テキストデータ符号化部
309
PC
101
バイナリ
XML

【図５】


START
テキストXMLを指定
S102
部分スキーマと参照する
データ型名を選択
S103
スキーマを単純化
S104
XPath式による選択
S105
テキストXMLを解析
S106
S107
全ての
テキストXMLを
符号化したか?
YES
NO
S108
解析中のテキスト
XMLはXPath式にマッチする
テキストデータを含むか?
NO
S109
符号化
YES
マッチしていないデータを符号化
S110
S111
マッチした
テキストデータは
属性値か?
NO
YES
属性名をタグの名前として
持つ開始タグを符号化
S112
マッチしたデータを逐次解析し、
選択したスキーマを用いて子要素化
S113
属性名をタグの名前として持つ
終了タグを符号化
S114
S115
マッチしたデータを
逐次解析し、選択した
スキーマを用いて子要素化
END

【図６】



【図７】

```
<grammar xmlns="http://www.w3.org/2000/svg">
 <define name="SVG.PathData.datatype">
  <list>
   <oneOrMore>
    <choice>
     <ref name="moveType"/>
     <ref name="lineType"/>
    </choice>
   </oneOrMore>
   <ref name="closepathType"/>
  </list>
 </define>
 <define name="moveType">
  <group>
   <choice>
    <value>M</value>
    <value>m</value>
   </choice>
   <data type="float"/>
   <data type="float"/>
  </group>
 </define>
 <define name="lineType">
  <group>
   <choice>
    <value>L</value>
    <value>l</value>
   </choice>
   <data type="float"/>
   <data type="float"/>
  </group>
 </define>
 <define name="closepathType">
  <group>
   <optional>
    <value>z</value>
   </optional>
  </group>
 </define>
</grammar>
```

【図８】

(a) //path@d

(b)

```
  <svg      xmlns="http://www.w3.org/2000/svg">
0x00 0x10  0x02 0x11 0x20

    <path      fill="red"          stroke="blue"      stroke-width="3">
  0x00 0x12   0x02 0x13 0x21   0x02 0x14 0x22    0x02 0x15 0x23

     <d>
   0x00 0x16

         <M>         100         100         </>
      0x00 0x17   0x03 0x24   0x03 0x24   0x01

         <L>         300         100         </>
      0x00 0x18   0x03 0x25   0x03 0x24   0x01

         <L>         200         300         </>
      0x00 0x18   0x03 0x26   0x03 0x25   0x01

         <z>       </>
      0x00 0x19   0x01

   </>
   0x01
 </>
 0x01
```

【図９】

符号	イベントの種類
0x00	開始タグ
0x01	終了タグ
0x02	属性
0x03	要素内容

【図１０】

符号	サイズ	名前
0x10	3	svg
0x11	4	xmlns
0x12	4	path
0x13	4	fill
0x14	6	stroke
0x15	12	stroke-width
0x16	1	d
0x17	1	M
0x18	1	L
0x19	1	z

【図１１】

(a)

符号	データ型	サイズ	値
0x20	0x30	26	http://www.w3.org/2000/svg
0x21	0x30	3	red
0x22	0x30	4	blue
0x23	0x30	1	3
0x24	0x35	4	100
0x25	0x35	4	300
0x26	0x35	4	200

(b)

符号	データ型
0x30	string
0x31	base64
0x32	short
0x33	int
0x34	long
0x35	float
0x36	double

フロントページの続き

(72)発明者　内田　均
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

Ｆターム(参考) 5B109 NH08