RDFデータモデルに基づくデータストアssdb

ssdb: Data Store Based on RDF Data Model

川　辺　治　之

要　約　運用稼働させてデータを蓄積しつつ継続的に機能拡張・改善するという方式でシステム開発がすすめられるケースが増えている．このような「永遠のβ版」に適したデータストアとしてssdbを設計・実装した．ssdbはそれぞれの管理対象に対して個別の属性を定義したり，後からそれらを追加・変更したりすることができる．ssdbはこれを実現するためにRDFのデータモデルを採用する．ssdbは，RDFデータの格納および問合せ言語SPARQLを効率よく実行するために，標準的な索引機能の上にRDF項管理モジュールおよびRDFステートメント管理モジュールを実装する．RDF項管理モジュールではすべてのRDF項を統一した内部形式を用いて管理し，RDFステートメント管理モジュールではRDFステートメントを複数の索引で管理することで，SPARQL問合せに対する結果を生成する．

Abstract　Various systems are successively extended and improved, collecting data from the production environment simultaneously. They are often denoted as “perpetual beta.” ssdb is a data store suitable for such systems, easy to define individual attributes for each record and easy to add or modify them later. ssdb stores data as RDF statements and executes SPARQL query language. It is implemented by the RDF term management and the RDF statement modules over simple indices. The former converts all RDF terms with various data types into internal representations in a unified way, and the latter stores RDF statements using multiple indices to execute SPARQL queries efficiently.

1. はじめに

機を逸せずに業務の要求に対応するために，また実際にシステム化してみなければわからない問題を早期に発見し解決するために，システム開発に対する考え方が変化してきている．それは，最初にシステム化の範囲・対象を厳密に規定してからシステム開発をはじめるのではなく，まずは見えている範囲でシステムを開発し，それを運用稼働させてデータを蓄積しつつ継続的に機能拡張・改善をすすめていくという方式であり，しばしば「永遠のβ版[1]」と呼称される．この過程において，データを収集する範囲が広がったりデータの構造が変わったりする場合がある．このような場合もそれまでに収集・蓄積したデータは捨てずに利用したいが，新しいデータ形式に無理に合わせようとすると，不整合が生じたり無駄なデータ格納領域が必要になったりする．これに対して，収集する個々の対象ごとに個別の属性の定義および後から追加・変更ができ，それらの属性に対する条件を指定して効率よく検索が実行できるデータストアがあれば，これらの問題を解決することができる．

しかし，このようなデータストアを広く一般に使われているDBMSを用いて効率よく実現するのは困難である．関係データベースを用いてこのようなデータストアを実現する場合，対象をレコードによって表現して次のいずれかの方式を用いるのが一般的である．

方式 1：すべての属性それぞれに対応する列をもつ（一つの）表を定義する．

方式 2：属性のもち方から対象をいくつかの種に分類し，種ごとに一つの表を定義する．

方式 3：属性の集まりを一つまたは固定個の列に格納する表を定義し，その列にそれぞれの属性をどう格納するかはアプリケーションに任せる．（たとえば，属性とその値の組の集合を XML で表現したものを一つの列に格納する．あるいは，それぞれの列がどの属性を表しているかを別の表で管理する．）

方式 1 では，この表にそれぞれのレコードが使用しない列が多数含まれ，また一つのレコードに新しい属性を追加する場合でも，スキーマの更新が必要となる．方式 2 でも，属性の追加・変更があるとスキーマの更新または新しい表の定義が必要となる．すると，複数の表を検索してマージする処理が必要となり，効率よく検索することが困難となる．方式 3 では，対象のデータ構造の一部がアプリケーションに埋め込まれて表現されていて，データとプログラムの独立性を損ない，個々の属性を効率よく検索することがむずかしい．

また，データストアとして XML データベースを用いる場合も，時間経過によってデータの構造が変化した場合には DTD または XML スキーマの変更を伴い，それまでに蓄積したデータの変換が必要となる．

そこで，対象ごとの個別の属性（とその値の対）の追加・削除・更新およびこれらの属性に対する条件を指定した検索を効率よく実行できるデータストアとして ssdb（Semi-Structured DataBase）を設計・実装した．ssdb のデータモデルおよび問合せ言語には，World Wide Web Consortium（W3C）が標準化をすすめている RDF（Resource Description Framework）のデータモデル[2]およびその問合せ言語 SPARQL[3]を採用した．RDF はインターネット上のさまざまな資源の「意味」を定式化して記述することを目的として，資源ごとに個別の属性を保持できることや標準的な属性を規定していることから，ssdb が管理対象とするデータを表現するのに適している．ただし，RDF は資源に対する情報（メタデータ）を表現することを主目的としているが，ssdb はメタデータだけでなく実世界で生じる事象についてもイベントデータとして管理し，さまざまなアプリケーションからそのデータを利用することを目的とする．

本論文の構成は次のとおりである．2 章では，RDF のデータモデルおよびそれに対する問合せ言語 SPARQL の概要を示す．3 章では，ssdb のアーキテクチャを示し，4 章では，ssdb における RDF データの表現方法を説明する．5 章では，SPARQL 問合せを効率よく実行するための索引付けの方式を提示する．6 章では，クライアントアプリケーションとのインターフェースについて述べる．そして 7 章では，ssdb の適用事例と今後の課題を述べる．

2. RDF の概要

この章で述べる RDF のデータモデルおよび SPARQL 問合せ言語の概要は，次章以降で ssdb の実装を示すのに必要となる部分のみにとどめる．

2.1 RDF のデータモデル

RDF データは一つ以上の RDF ステートメントにより構成される．それぞれの RDF ステートメントは，主語，述語，および目的語の三つの要素で構成される．主語は IRI（Internationalized Resource Identifier）または空白ノードを用いてその IRI で指し示される対象を一意に

識別する．空白ノードは，ある RDF データの範囲の中だけで識別可能であるが，IRI のような一意に識別することのできる外部表現をもたない．

述語は，IRI を用いて主語が指し示す対象がもつ属性を表現する．目的語はその対象がもつ属性の値を表す．標準的に使用される述語は，RDF 語彙集合（ボキャブラリ）として W3C またはその他の団体により標準化がすすめられている．この標準化された RDF 語彙集合を使用することで，データが表す意味を定式的に記述できる．

目的語には RDF 項が使用される．RDF 項は，IRI，空白ノード，リテラル（単純リテラル，言語タグ付きリテラル，型付きリテラル）のいずれかである．型付きリテラルとしては，整数型や日付型など W3C が規定する XML データ型が使用できる．同一の IRI を主語とする一つ以上の RDF ステートメントによって，その主語が指し示す対象の性質や状態を表現する．ラベル付き有向グラフで RDF ステートメントを表記する場合，有向辺が述語，有向辺の始点が主語，有向辺の終点が目的語となる．そして，主語および目的語が IRI または空白ノードの場合は頂点を楕円で表記し，リテラルの場合は長方形で表記する．ラベル付き有向グラフで表記した RDF ステートメントの例を図 1 に示す．

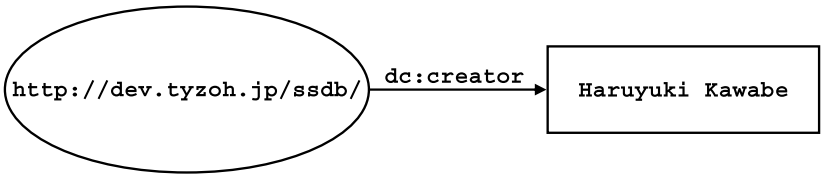

図 1　RDF ステートメントの例

この図において，有向辺に付けられたラベル "dc:creator" が述語，この有向辺の始点 "http://dev.tyzoh.jp/ssdb" が主語，そして有向辺の終点 "Haruyuki Kawabe" が目的語となる．この例は，主語で指し示される（インターネット上の）資源の作成者が "Haruyuki Kawabe" であることを表す RDF ステートメントである．

目的語として別の IRI または空白ノードを使用することで，RDF ステートメントを組み合わせて複雑な構造を表現することができる．RDF ステートメントの集まりに対して，それぞ

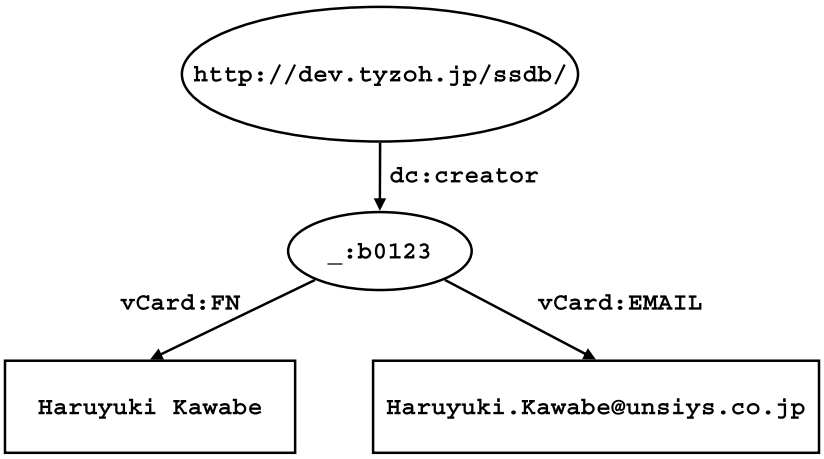

図 2　RDF グラフの例

れのリテラルおよび同一の IRI・空白ノードを頂点とし，それぞれの述語を有向辺として得られるグラフを RDF グラフと呼ぶ．三つの RDF ステートメントからなる RDF グラフの例を図 2 に示す．"dc:creator" とラベル付けされた有向辺の終点となる頂点は空白ノードである．

2. 2　SPARQL 問合せ言語

SPARQL は RDF データに対する問合せ言語である．SPARQL 問合せには，対象となる RDF データが指定した条件を満たすかどうかを調べる ASK 文，対象となる RDF データから指定した条件のデータを取り出す SELECT 文，対象となる RDF データから新たな RDF ステートメントの組を生成する CONSTRUCT 文，対象となる RDF データに含まれる指定した資源に関する情報を表示する DESCRIBE 文の 4 種類がある．前三者は，いずれも RDF ステートメントに対するパタン（トリプルパタン）の組合せおよびそれらに対する制約式によって問合せ結果を得るための条件を指定する．トリプルパタンは，RDF ステートメントの主語，述語，目的語のいくつかを変数で置き換えたものである．問合せに含まれる変数の集合から RDF 項（IRI，空白ノード，および RDF リテラルを合わせたもの）への部分関数を解写像と呼ぶ．解写像は，変数とそれが写像される RDF 項の対の集合によって表現する．問合せの条件に指定されたトリプルパタンおよび制約式に対して，それらに含まれる変数を解写像によって写される RDF 項で置き換えると，（変数を含まない）RDF ステートメントと式が得られる．こうして得られたすべての RDF ステートメントが対象となる RDF データに含まれ，すべての式が成立するとき，この解写像をこの条件に対する解という．（厳密にいえば，SPARQL 問合せの条件に OPTIONAL 節や UNION 節を含む場合はこの限りではない．）SPARQL 問合せの実行は，対象となる RDF データから問合せの条件に含まれる個々のトリプルパタンに対する解を生成する処理と，その解である解写像を合成して問合せの結果を生成する処理にわかれる．解写像の合成は，同じ変数に対応する値が一致する解写像を組み合わせて新たな解写像を生成する処理となる．解写像を用いた SPARQL 問合せの条件中の変数の置換および解写像の合成の例を図 3 に示す．

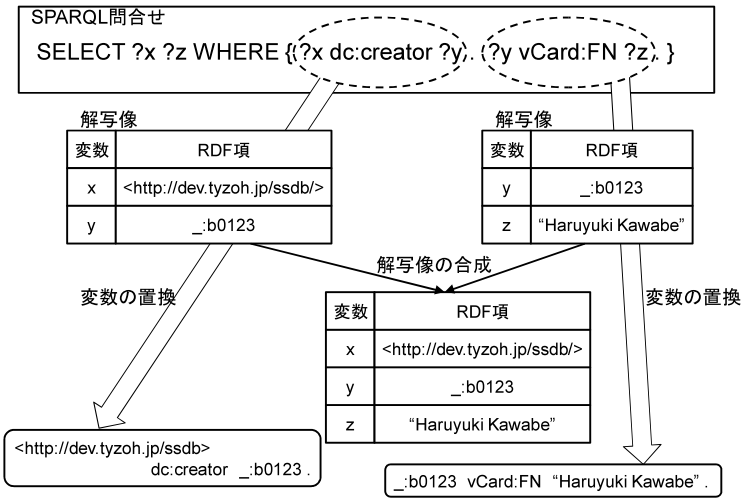

図 3　SPARQL 問合せの例と解写像の合成

3. ssdb のアーキテクチャ

ssdb は，複数のクライアントアプリケーションが共有する RDF データを同時に更新または検索できるようにクライアントサーバ型で動作する．RDF データはサーバプロセスで管理し，クライアントアプリケーションは通信インターフェースを介してこの RDF データの更新および検索を行う．サーバプロセスは，索引を用いて SPARQL 問合せの条件に適合する RDF ステートメントを永続媒体から効率よく取り出して解となる解写像を生成し，メモリ上の処理でそれらを合成して問合せの結果とする．

ssdb サーバは図 4 に示すモジュールから構成される．

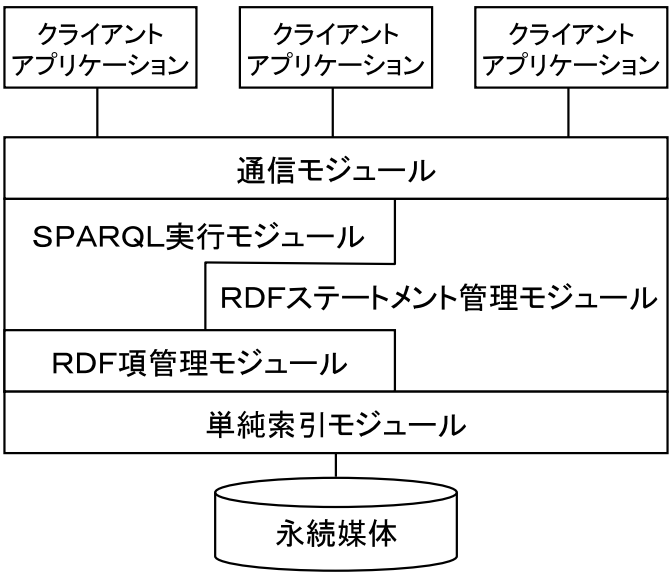

図 4　ssdb のモジュール構成

- 単純索引モジュール

  RDF 項管理モジュールと RDF ステートメント管理モジュールの下位に位置し，キー・値対を用いた永続的索引に対する追加・削除・検索機能を提供する．永続媒体に RDF データを格納する機能を単純索引モジュールとして切り出すことで，標準的な索引機能をもつモジュールによる置き換えを可能とし，分散索引や大規模索引アルゴリズムの適用を容易にする．

- RDF 項管理モジュール

  RDF 項の外部表現と内部表現の対応を管理する．単純索引モジュールで実装された索引を使用して，外部形式と内部形式の間の変換を行う．詳細は 4 章「RDF 項の表現方式」を参照のこと．

- RDF ステートメント管理モジュール

  入力された RDF ステートメントの追加・削除を行う．RDF 項管理モジュールを使用して入力された RDF ステートメントを内部形式に変換し，それを単純索引モジュールで実装された索引に追加または索引から削除する．索引を用いた RDF ステートメントの格納方式については 5 章「SPARQL 問合せに適した索引付け」を参照のこと．

- SPARQL 実行モジュール
  入力された SPARQL 問合せをパースして内部形式に変換し，RDF ステートメント管理モジュールを使ってトリプルパタンに対する解を生成し，解写像の合成によって問合せの結果を生成する．
- 通信モジュール
  クライアントから HTTP などの通信プロトコルを介して RDF ステートメント追加・削除および SPARQL 問合せを受け付け，クライアントに SPARQL 問合せの結果を返す．利用するアプリケーションによってそれに適した通信モジュールを選択することができる．詳細は 6 章「クライアントアプリケーションからのインターフェース」を参照のこと．

4. RDF 項の表現方式

ssdb は，RDF データおよび SPARQL 問合せの入力時にすべての RDF 項に一意となる識別子を割り当て，その識別子を使用して RDF ステートメントを表現する．これにより RDF 項の外部表現をそのまま索引付けするのに比べて RDF ステートメントを格納する索引のサイズが小さくなり，解写像を効率よく合成することができる．

任意の RDF 項が RDF ステートメントの目的語となりうるので，目的語として RDF 項を格納する領域は任意の型のデータを保持できる必要がある．ssdb は，型を識別する type とその型の中で一意にデータを識別する key の対（type，key）によってすべての RDF 項を表現する．これを内部表現と呼ぶ．それぞれのデータ型の中で RDF 項を一意に識別するために，内部的な発番機構を用いてそれぞれの RDF 項に対して key を生成する．それぞれのデータ型に対して割り当てた型識別子を表 1 に示す．トリプルパタンに含まれる変数，および解写像における「値無し」を表す NULL 値に対しても型識別子 0 とする同じ内部表現を用いる．

表 1　データ型と型識別子

データ型		型識別子
変数およびNULL値		0
datatype		1
空白ノード		2
IRI		3
言語タグ		10
型付きリテラル	string	4
	boolean	11
	integer	12
	⋮	⋮
単純リテラル		-1
言語タグ付きリテラル	Lang=EN	-2
	⋮	⋮

また，RDF ステートメントの主語および述語も IRI（型識別子 3）として同じ内部表現を用いることで，IRI の外部表現を用いるのに比べて RDF ステートメントを格納する索引のサイズを小さくできる．これによって，内部表現同士の比較を用いて解写像を合成できるので効率よく問合せ結果を得られる．ただし，問合せ結果を出力するときには内部形式から外部形式へ変換するため結果件数に比例したオーバーヘッドが生じるが，トリプルパタンに対する解の生

成と解写像の合成の効率を重視した．

それぞれの RDF 項の外部表現と内部表現を一対一対応させるために，アクセサと呼ぶ対応表を使用する．アクセサは，内部表現の key をキーとし外部表現を値とする索引，および外部表現をキーとし内部表現の key を値とする索引により実装する．前者は内部表現から外部表現への変換に使用し，後者は外部表現から内部表現への変換に使用する．ただし boolean，integer，float，double，dateTime などの型については，それらの値を表すビット列を key とすることで，外部表現と内部表現の間の変換に索引を必要としない．このように索引を使用しないデータ型を即値型と呼ぶ．データ型ごとのアクセサを図 5 に示す．この図の中で左側にある表は，データ型とアクセサの対応を管理する．ただし，図中の点線で表記した部分は即値型に対するアクセサとなり索引を使用しない．

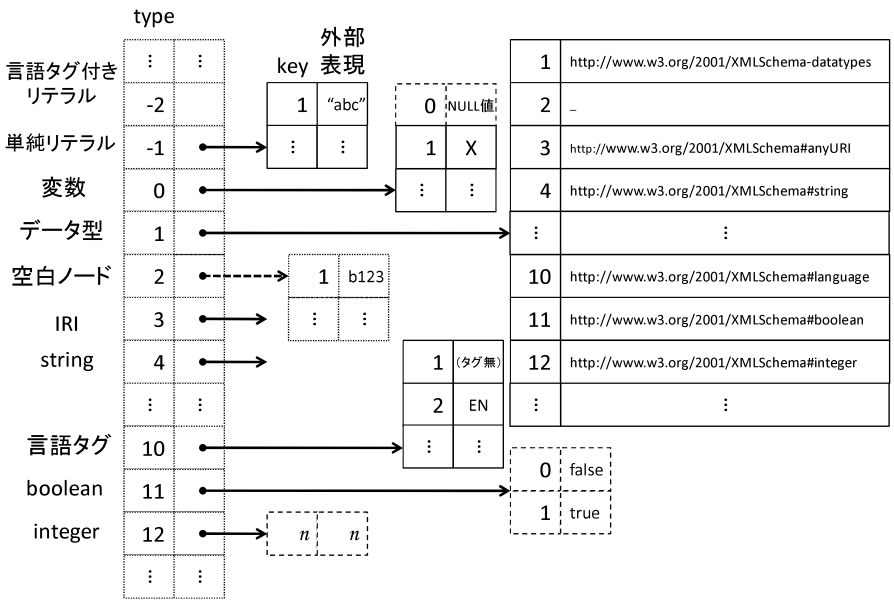

図 5　内部表現と外部表現の対応付け

また，データ型と言語タグにも型識別子（それぞれ型識別子 1 および 10）を割り当てることで，型付きリテラルのデータ型および言語タグ付きリテラルの言語タグもアクセサを用いて管理する．これによって，新しい型のリテラルや言語タグをもつリテラルを読み込んだ場合にも，それらを一意に識別する型識別子を割り当てる．

このようにすべてのデータを同じ形式の内部表現に変換することで，5 章で示す SPARQL 問合せ実行の際の解写像の生成および合成が，索引の標準的な機能およびバイト列の同一性判定によって実現できる．ただし，文字列やリテラルの辞書式順序を用いた大小比較や部分文字列一致を実行するには外部表現を必要とするため，アクセサを用いて外部表現に変換するオーバーヘッドがある．

5. SPARQL 問合せに適した索引付け

ssdb は，内部表現を使用して RDF ステートメントを索引に格納することで，問合せに含まれるトリプルパタンおよび制約式から効率よく解を生成する．ただし，トリプルパタンのうち

利用頻度の低いトリプルパタンに対する索引は作成せず他の索引から解を生成することで，RDF ステートメント登録時の索引作成のオーバーヘッドが低減する．

索引そのものに RDF ステートメントを格納することで，解写像を生成する際に索引から変数に対応する値を直接得ることができ，トリプルパタンに対する解を効率よく生成できる．トリプルパタンは，RDF ステートメントの主語，述語，目的語のどれを変数とするかによって $2^3 = 8$ 通りの組合せがある．これら 8 通りのそれぞれについて，トリプルパタン中で RDF 項である要素をキーとし変数である要素を値とするような索引があれば，そのトリプルパタンに対する解を効率よく生成できる．しかし，RDF ステートメントの追加・削除の際にこれらの索引を更新することになるので，索引の数に比例した負荷がかかる．そこで，8 通りのうち利用頻度の低い索引は作成せず，それらに対応するトリプルパタンに対する解は，他の索引を用いて生成する．

SPARQL 問合せで指定される条件のうち典型的なものは，述語が IRI となっている次の 3 通りおよびその組合せである．

- 指定された属性（述語）に対して，その属性をもつ識別子（主語）とその属性値（目的語）の並びを取得する
  例）SELECT ?r ?c WHERE { ?r dc:creator ?c . }
- 指定された属性（述語）とその値（目的語）に対して，そのような属性値をもつ識別子（主語）の並びを取得する
  例）SELECT ?r WHERE { ?r dc:creator ″豊洲太郎″ . }
- 指定された識別子（主語）と属性（述語）に対して，その識別子のその属性の値（目的語）を取得する
  例）SELECT ?c WHERE { <http://www.tyzoh.jp/> dc:creator ?c . }

これらの条件指定を関係データベースに対する問合せと対比させてみると，問合せ中で述語に相当する列名を明示的に指定することからも妥当な仮定である．これらの問合せに対応するトリプルパタンは表 2 の網掛で表記した 3 種類となり，これらに対して効率よく解を生成するための索引を作成する．

表 2　トリプルパタンの組合せ

主語	述語	目的語
変数	変数	変数
RDF項	変数	変数
変数	RDF項	変数
RDF項	RDF項	変数
変数	変数	RDF項
RDF項	変数	RDF項
変数	RDF項	RDF項
RDF項	RDF項	RDF項

また，使用する単純索引モジュールは次の検索方式を提供していることを前提とする．

1）　キー指定
　指定したキーに対して，それに結び付けられた値の並びを取得する．

2）　キー範囲指定
　指定した範囲に含まれるキーそれぞれに結び付けられた値の並びを取得する．

3) 全件探索

索引を全件探索し，条件に合致するキーに結び付けられた値の並びを取得する．

4) 値比較

上記 1)，2)，3)それぞれの場合に，取得した値が条件を満たしているかどうかの判定を行う．

5) 値範囲指定

キー指定の検索の際に，指定されたキーに結び付けられた値のうち指定した範囲に含まれるものの並びを取得する．

さらに，一般には，キー指定，キー範囲指定，全件探索の順に効率よく値を取得することができるものとする．例えば索引が b-tree を用いて実装されている場合には，これらの前提条件を満足する．

この前提のもとで，表 3 にあげる二つの索引に RDF ステートメントを格納する．

表 3　RDF ステートメントを表現する索引

キー	値
主語＋述語	目的語
述語＋目的語	主語

これらの索引では，4 章で述べた内部表現を用いてキーおよび値を保持する．また，述語は常に同一の type（URI 型）をもつ内部表現となるので，索引のキーには key だけを格納することで索引のサイズを縮小させ，検索の際のキーの冗長な比較を取り除いた．

前述の 8 通りのトリプルパタンに対して，それぞれ表 4 にあげる索引と検索機能を用いて解を生成する．網掛で表記した行が典型的に使用されるトリプルパタンとなる．

表 4　トリプルパタンの検索に使用する索引と検索機能

トリプルパタン			使用する索引		使用する検索機能
主語	述語	目的語	キー	値	
変数	変数	変数	主語＋述語	目的語	全件探索
RDF項	変数	変数	主語＋述語	目的語	キー範囲指定
変数	RDF項	変数	述語＋目的語	主語	キー範囲指定
RDF項	RDF項	変数	主語＋述語	目的語	キー指定
変数	変数	RDF項	主語＋述語	目的語	全件探索＋値比較
RDF項	変数	RDF項	主語＋述語	目的語	キー範囲指定＋値比較
変数	RDF項	RDF項	述語＋目的語	主語	キー指定
RDF項	RDF項	RDF項	述語＋目的語	主語	キー指定＋値比較

また，トリプルパタンの目的語が変数となる場合，問合せの条件中に制約としてその目的語に対する範囲やデータ型が指定されることがある．制約が即値型に対する範囲指定となる場合，即値型の内部表現（type，key）の key そのものがその数値や時刻の値を表すので，指定されたデータ型が type でかつ key が指定された範囲に含まれるものを選び出す処理となる．制約がデータ型指定となる場合は，指定されたデータ型に type が一致するものまたは一致しないものを選び出す処理となる．これらの処理は，索引のキー範囲指定または値範囲指定として効率よく検索することができる．これらの場合に使用する索引と検索機能を表 5 に示す．網掛で表記した行が典型的に使用されるトリプルパタンとなる．

表5　範囲指定検索となる場合の索引と検索機能

トリプルパタン			使用する索引		使用する検索機能
主語	述語	目的語	キー	値	
変数	変数	範囲指定	主語＋述語	目的語	全件探索＋値比較
RDF項	変数	範囲指定	主語＋述語	目的語	キー範囲指定＋値比較
変数	RDF項	範囲指定	述語＋目的語	主語	キー範囲指定
RDF項	RDF項	範囲指定	主語＋述語	目的語	キー指定＋値範囲指定

索引の範囲指定に変換できない制約については，索引から値を取り出した時または解写像の合成時に制約を満たさない解写像を取り除く．

6.　クライアントアプリケーションからのインターフェース

ssdb は RDF ステートメントの追加・削除および SPARQL 問合せを受け付ける通信モジュールとして，次の三つの方式を提供する．アプリケーションは利用形態に応じて，それに適したインターフェースを介して ssdb サーバプロセスにアクセスすることができる．

- HTTP プロトコル

  サーバプロセスのフロント部分に Web サーバ（apache など）を配置し，Web サーバの常駐型 CGI（fast CGI）アプリケーションとして ssdb サーバプロセスを稼働させることで，Web サーバが標準的に提供する暗号化，認証，流量制御等の機能を利用することができる．java や javascript には HTTP 要求を送信するモジュールが標準的に用意されているので，クライアントプログラムを手軽に実装できる．また RDF および SPARQL の表現形式としてそれぞれ RDF/XML[4]および SPARQL 問合せ結果の XML 形式[5]を採用しているので，他の RDF ツールとの相互運用性も高い．

- 独自プロトコル

  通信モジュールで，クライアントとサーバの間の単一の TCP セッションを仮想的に多重化し，切断された TCP セッションの再確立を自動的に行う．また，単純化した独自データ形式で RDF ステートメントを表現する通信プロトコルを実装し，RDF/XML 形式のパージングを不要とした．ssdb サーバとクライアントアプリケーションの間の通信量が多く，効率が要求される場合に適している．

- ライブラリ形式

  アプリケーションプログラムに ssdb サーバプログラムをライブラリとしてリンクする．アプリケーションプログラムからライブラリ呼出しにより RDF データの参照および更新を行うので，クライアント－サーバ間の通信のオーバーヘッドがない．ただし，この方式では RDF データを格納するデータストアを共有して複数のアプリケーションプログラムから同時にアクセスすることはできない．

RDF のデータモデルと多くの共通点をもつデータモデルを採用している ucode 解決[6]では，ucode 関係[7]という主語－述語－目的語の三つ組を用いて対象の属性や状態を表現する．ただし ucode 関係における主語は対象を指し示す ucode を含む URI である．ucode 解決プロトコルは，この ucode 関係の集まりによって表現されたデータに対する問合せや ucode 関係の追加・削除・更新を行うプロトコルである．この ucode 解決プロトコルを TCP ストリームとし

て受け取り処理するインターフェースを ssdb に実装した．このインターフェースを使うことで，ssdb を ucode 解決サーバとして使用することができる．ucode 解決プロトコルにおける問合せは，SPARQL 問合せと同じ内部形式に変換し処理する．またユーザ定義問合せとして SPARQL 問合せも使用できる．

7. お わ り に

データを収集・蓄積しつつ明らかになった課題を解決するための機能拡張・改善に伴い，管理対象ごとに保持する属性が変遷していくシステムのデータストアとして ssdb を使用し，その有効性を検証した．これらのシステムでは，それまでに蓄積したデータを破棄・変更することなく，継続的にシステムの機能拡張・改善を実施できることが確認できた．このようなシステムのうち，実証実験および本番稼働しているシステムとして次のものがある．

- 次世代文書管理システム tacoPot[8]

  日本ユニシス(株)総合技術研究所先端技術部にて研究開発を行っている tacoPot は，登録された文書に対して，登録者だけでなく利用者の視点からタグ付けを行うことで，様々な切り口による検索を可能とするシステムである．ssdb を用いてこの文書に付与されたタグおよび文書のメタデータ（著者，タイトル，登録日など）を管理する．本番運用をしながら利用者の声を反映して機能拡張・改善を行う上で，RDF のデータモデルの柔軟な表現形式が有効に機能している．

- マルチコード相互運用電子タグ実証事業[9]

  複数の電子タグ仕様の相互運用および個々の仕様に依存しないアプリケーションを開発するためのプラットフォームを設計・実装し，実証実験を行った．このマルチコードプラットフォームにおいて，電子タグを用いて取得されるイベントデータ（いつ，どこに設置されたタグリーダで，どの電子タグを読み取ったか）を管理するデータストアとして ssdb を採用した．

- 健康情報管理システム

  生活習慣改善に役立て，特定の症例との相関関係の分析を行うために，各人の健康情報を収集・蓄積するシステムの実証実験の中で，健康情報の一部を RDF データの形で蓄積することで ssdb の機能評価を行った．

これらのシステムでは，文書，イベント，または健康情報といった対象ごとに個別の属性を追加・削除・更新することができ，柔軟な問合せが行えるデータモデルの有効性が検証できた．また，このような柔軟なデータモデルを採用したデータストアの実装技術が実用に供しうるものであることを確認できた．今後は，分散システム対応や大規模索引アルゴリズムの採用などにより，大量の RDF データを蓄積し，さまざまなアプリケーションから利用することのできるデータストアを実現していきたい．

最後に，半構造化データベース研究開発の発端となり ssdb の前身でもある Rinza RDF Repository 開発メンバー，および ssdb の修正・移植・テストや ssdb をデータストアとして用いたアプリケーションの開発を行い有益なフィードバックをくださった日本ユニシス(株)総合技術研究所先端技術部の諸氏に感謝の意を表したい．

参考文献
［1］ Tim O'Reilly, "What Is Web 2.0", 2005.9.30, http://oreilly.com/pub/a/web2/archive/what-is-web-20.html
［2］ World Wide Web Consortium, "Resource Description Framework (RDF): Concepts and Abstract Syntax", 2004.2.14, http://www.w3.org/TR/rdf-concepts/
［3］ World Wide Web Consortium, "SPARQL Query Language for RDF", 2008.2.15, http://www.w3.org/TR/rdf-sparql-query/
［4］ World Wide Web Consortium, "RDF/XML Syntax Specification (Revised)", 2004.2.10, http://www.w3.org/TR/rdf-syntax-grammar/
［5］ World Wide Web Consortium, "SPARQL Query Results XML Format", 2008.1.15, http://www.w3.org/TR/rdf-sparql-XMLres/
［6］ T-Engine フォーラム，「ucode 解決プロトコル（標準版）」，Ubiquitous ID Center Specification, 910-S211-0.00.04/UID-CO00008-0.00.04, 2006.10.12, http://www.t-engine.org/japanese/archives/UID-CO00008-0.00.04.pdf
［7］ T-Engine フォーラム，「UCR/XML: XML による UCR graph のシリアライズ」，Ubiquitous ID Center Specification, 940-S102-0.00.17/UID-CO00027-0.00.17, 2006.12.25, http://www.t-engine.org/japanese/archives/UID-CO00027-0.00.17.pdf
［8］ 佐藤一広，「タグを用いた企業内情報共有」，ユニシス技報，日本ユニシス，Vol.29 No.2 通巻 101 号，2009 年 8 月
［9］ 日本ユニシス株式会社，東京大学 21 世紀 COE「次世代ユビキタス情報社会基盤の形成」，慶應義塾大学 SFC 研究所，ユビキタス ID センター，Auto-ID Lab. Japan，「平成 18 年度エネルギー使用合理化電子タグシステム開発調査事業マルチコード相互運用電子タグ実証実験報告書」，2007 年 2 月，http://www.meti.go.jp/policy/it_policy/tag/2006_Mcode_Report.pdf

（参考文献に挙げた URL は 2009 年 8 月 11 日時点で存在を確認）

執筆者紹介 川 辺 治 之（Haruyuki Kawabe）
1985 年日本ユニシス(株)入社．Lisp マシン・UNIX 等オープン系基盤ソフトウェアの開発・保守，ASP 事業の企画・運営，半構造化データベースの研究開発に従事．現在，ICT サービス本部 SASTIK サービス推進部に所属．