存在従属グラフからRESTful Webサービスの生成
Generating RESTful Web Services from Existence Dependency Graph

井田明男† 金田重郎† 森本悠介†
Akio Ida† Shigeo Kaneda† Yusuke Morimoto†
†同志社大学大学院・理工学研究科
† Graduate School of Science and Engineering, Doshisha University, Kyotanabe, Kyoto 610-0394, Japan

要旨
本提案は，強靭な骨格と柔軟な相互接続性を併せ持つ業務アプリケーションの構築を狙いとして，ドメインモデルとしての存在従属グラフから，グラフ上の要素にアクセスするための Web サービスを生成し，適用業務から呼び出して使えるようにする．生成される Web サービス群はエンティティ間の 4 つの関係を参照することにより REST LEVEL3 を満たすようにした．すなわち，HATEOAS の概念を導入している．適用業務がグラフの要素にアクセスする際は，統一的なインターフェースを介してのみアクセスするため，おのおのの適用業務は相互接続性に優れたものとなる．

1. はじめに

情報システム開発の現場においては，経営環境の変化にシステムが機敏かつ柔軟に追随できる方策を求めて，マイクロサービス・アーキテクチャ（MSA）に関心が寄せられている[1]．MSA は業務アプリケーションを独立して配置可能なサービスの組み合わせとして設計・実装する方法を指すものとして，ここ数年で急速に認知されている．社内外のさまざまなシステムが公開するサービスを組み合わせることによって，新たな業務アプリケーションや，より粒度の大きなサービス（以下ではこれらを指して「適用業務」と記す場合がある）を実装していく開発スタイルの総称である．MSA の実現技術は多様であるが，REST（Representational State Transfer）アーキテクチャ[2]はその実現技術の代表格であるといえよう．このような適用業務を MSA を前提として設計および実装を行うプロセスは，アジャイル開発と非常に親和性が高い．なぜならば，適用業務の切り出しが上手くできれば開発対象を限定してのインクリメンタルな開発が可能となるからである．

しかしながら，実際には局所最適化された適用業務が量産され，それらの相互接続性が著しく低いといった失敗例が多く報告されている[3]．これは，大規模な業務システムの場合，最初に業務全体を適切な粒度にドメイン分割したり，ドメイン毎にモデルを構築し，ドメイン間のインターフェースを仕様化するといった全体的な視点での設計成果物が不可欠であるにもかかわらず，それらが十分に実施されて来なかったことに起因する．そのため，最近では，アジャイル型開発にウォータフォール型開発の利点を採り入れようとする動きがある．

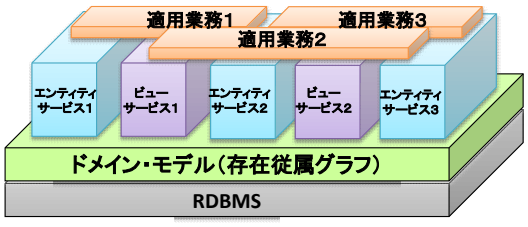

図 1：適用業務，サービス，ドメインモデルの階層

例えば，Disciplined Agile Delivery[3]では UP（Unified Process）[4]と同じように方向付けフェーズを設けており，全体の視点でのモデリング機会が確保されている．このような開発プロセスでは，全体の視点で作成したモデルを個別の視点で利用可能な成果物に展開するためのガイドラインが不可欠である．しかしながら，現時点では全体的な視点で作成したドメインモデルからサービスを適切に構造化し，API の仕様を公開していくための体系的なガイドラインが示されているとは言い難い．

そこで，本稿では，強靭な骨格と，柔軟な相互接続性を併せ持つ適用業務の構築を狙いとして，ドメインモデルとしての存在従属グラフから REST サービス群を生成しそれらを利用してのエンタープライズ・アジャイル開発に資するためのツールと手法を提案する．**図 1** は提案手法が目指すアーキテクチャのイメージである．具体的には，全体的な視点で存在従属分析により構築した存在従属グラフを骨格とするデータベースを先ず構築する．次に，それを土台として粒度の細かいサービス群をツールで生成的に定義し，それらを呼び出す形で適用業務を実装していく．

以下，２章では，本提案に関連する背景と基礎知識について説明する．３章では，提案手法について説明する．４章では，開発現場における提案手法の適用結果について概要を報告し，本稿のまとめとする．

2. 背景と基礎知識

5.1. サービス指向

MSA は適用業務を相互接続性に優れたモジュラリティの高いサービスの組み合わせとして設計・実装していこうとする考え方である．かつて SOA（Service Oriented Architecture）が企業情報システム開発の分野において脚光を浴びていたが, SOA が要求する技術は「大 Web サービス」と呼ばれ, SOAP や WSDL など WS-*と称される一連の仕様群を前提とする[5].

一方，MSA は，Web の文化から生まれた軽量な Web API をサービス群の主な実現技術にするため，それらの連携に ESB のような重量級のミドルウェアを必要としない．なかでも REST は，ウェブのような分散システムのためのソフトウェアアーキテクチャのスタイルのひとつである. この語は Fielding が，ウェブについて書いた論文[2]で初めて紹介され，開発者の間で急速に認知され使われるようになった.

表 1 は Burke の挙げる REST アーキテクチャの原則[2][6]を要約したものである．REST アーキテクチャの原則に従うサービスは RESTful サービスと呼ばれる．表 1 に示した原則の中でも「制約されたインターフェース」は相互接続性のために筆者らが重視する原則であるが, 手作業で守り通すことは難しい.

表 1：RESTful アーキテクチャの原則と設計レベル

RESTfulアーキテクチャの原則	説明
アドレス可能なリソース	リソースは，RESTにおいて情報とデータの主要な抽象である．個々のリソースはURI（Uniform Resource Identifier）を通じてアドレス可能でなければならない(REST LEVEL1)
制約された統一インターフェース	リソースを操作するための明確なメソッドの小さなセットを使用する．URIには「アクション」パラメータを使用せず，WebサービスはPOST，GET，PUT，DELETEなどのHTTPのメソッドのみを使用する(REST LEVEL2)
表現指向	サービスとの対話には，そのサービスが提供する表現を使用する．1つのURIによって参照されるリソースは，複数のフォーマット（表現）を持つことができる
ステートレスな通信	すべてのHTTPリクエストが完全に分離している．クライアントがリクエストを送信する際には，サーバがリスエストを処理するために必要なアプリケーションの状態をすべて含めなければならない(REST LEVEL0)
アプリケーション状態エンジンとしてのハイパーメディア（HATEOAS）	サーバはリンク付きの表現を返すことで，最新の戻り値の結果に応じてクライアントが次に送信可能なリクエストについての示唆を与えることができる（REST LEVEL3）

5.2. アジャイル開発手法

初期のアジャイル手法は，開発単位を小さく切り出して，その単位毎に実装しながらデータベース構造や適用業務の仕様を探ることを旨とする．結果的に，適用業務毎に局所最適化されたデータベースが生み出されやすい．そのため最近では，エンタープライズ・アジャイル開発手法が注目されている．代表的なプロセスフレームワークとして 2 つが挙げられる．一つは Leffingwell が提唱する「スケールド・アジャイル・フレームワーク（Scaled Agile Framework：SAFe）」[7]，もう一つは Ambler が提唱する「ディシプリンド・アジャイル・デリバリ（Disciplined Agile Delivery）フレームワーク」[3]である.

これらのエンタープライズ・アジャイル開発では全体の視点と個別の視点が要求されるため，前者の視点で作成したモデルを後者の視点で利用可能な成果物に展開するためのガイドラインが不可欠である. しかしながら，現時点では全体的な視点で作成したドメインモデルからサービスを適切に構造化，公開していくための体系的なガイドラインが示されているとは言い難い.

5.3. アジャイル開発手法におけるドキュメント Swagger

Swagger は RESTful API の仕様書を自動生成するための一連のソフトウェア群である. API を提供するサービスにおいて，Swagger を利用して API の実装に必要な情報を追加すると，ブラウザで閲覧・実行

が可能な仕様書を生成することができる．公式サイトでは Swagger で生成された仕様書の見本が公開されている．

Swagger の特徴は実行可能な仕様書を提供することである．閲覧者は，リソースの一覧から詳しく参照したいリソースを選択すると，そのリソースに対して利用可能な操作（POST，GET，PUT，DELETE など）の一覧が表示される．さらに，仕様を参照するだけでなく，実際にサーバの API を呼び出して結果を表示する機能が備わっている．その場合，仕様書が展開してレスポンスの情報が表示される．かくして閲覧者は操作の内容を具体的に知ることができる．

Swagger は，REST API のリソースに関する情報を提供するサーバ側と，その情報を解析してリファレンスを表示するクライアント側のツールから構成される．クライアントは，サーバから Swagger Spec という JSON または YAML 形式のデータで API に関する情報を取得し，仕様書を生成する．これによって実際にサーバ側で動いているコードと乖離のない仕様書を常に提供することができる．しかしながら Swagger Spec を手作業で記述する場合は API を実装するのと同程度の作業工数が必要である．

3. 提案手法

そこで，業務要求の記述から存在従属グラフを構築し，Data Access RESTful Service（DARS）を生成して運用していくためのプロセスを提案する．このプロセスは単独で実施するものではなく，DAD などに準拠したエンタープライズ･アジャイル開発を実施する際に, 補完的に適用することを想定している．

5.4. エンティティの存在従属グラフを構築する

エンティティとは，業務ドメインで捕捉，蓄積，管理すべき重要な概念である．エンティティの存在従属グラフは，エンティティのインスタンスの存在期間の関係である存在従属性に着目して作成する有向グラフである．**図 2** は受注管理ドメインのグラフ例である．

存在従属グラフは UML クラス図の記法を流用しているため，静的な構造図に見えるが，実際はエンティティのインスタンスの存在期間に基づく依存関係を表現するので筆者らは動的モデルとしても位置づけている．日本語要求記述文からのエンティティの識別については文献[8]を，存在従属という概念，および存在従属グラフ作成の詳細については，文献[9]を，それぞれ参照されたい．

5.5. サービスの生成 - 存在従属グラフから DARS の原型定義を生成する

生成すべきサービスは, グラフ上に登場するそれぞれのエンティティのインスタンスを CRUD するためのサービス群であるため, サービスの数は単純に計算してもエンティティ数の 4 倍になる. 実際には, 照会（refer）の処理などにいくつかのバリエーションを持たせることになるためエンティティティ数の 8～10 倍程度になる．これはとても手作業では対応しきれない数である．

そこで，筆者らはプログラム（GenDARS と称する）を自作することにした ．GenDARS の入力は存在従属グラフの xml 表現である．そして，GenDARS の出力は，RDB の Create Table 文，Create View 文，DARS の原型定義の Scalatra のソース・コード群，および，DARS の原型定義を呼び出すクライアントの Sinatra（Ruby）のコードである. 図 2 の存在従属グラフをトレースしつつ, 適用業務の実現に必要な DARS はどのようなものであるかを確認した結果を**表 2** に示す．なお，出力として Scalatra／Sinatra を選んだのは，それらが「URI パタンに対して行うべき処理」という一定の形式でサービスを記述していける特性を持っていたからである．なお，Swagger Spec は Scalatra のコードから生成できる（**図 3**）.

5.6. 生成が必要なサービスを検討する

表 2 の検討結果から**図 4** ようなレイヤ構成が得られる．そして，これは，図 1 の階層とも一致する．適用業務がエンティティに対する検索サービスを呼び出す代わりに，関連，すなわち仮想表に対するサービスを呼び出す選択肢があることも示唆される．エンティティ，関連に対するいずれのサービスを呼び出しても目的が達成される場合は，関連に対するサービスを呼び出した方が目的に対して絞り込まれたインスタンスの一覧を取得できる．

5.7. 隣接行列を生成する

GenDARS 内部ではエンティティの存在従属グラフを読み込んで，隣接行列を生成する．存在従属グラフは有向グラフであるため生成すべき隣接行列は対角成分に対して非対称な正方行列となる．また，

関係の両端のエンティティを始点と終点の役割で区別する必要がある．隣接行列の行と列にそれぞれエンティティを割当て，各セルには，行のエンティティから列のエンティティに向かう関係の種類（無関係=0，属性従属関係=1，存在従属関係=2，参照関係=3，または汎化関係=4 のいずれか）を記入する．隣接行列を走査することより，生成すべき API と URI の命名が明らかになる．

5.8. エンティティ・サービスを生成する

エンティティ・サービスとは，存在従属グラフに登場するおのおののエンティティについてのクラスおよびインスタンス・スコープのサービス群である． 1 つのエンティティにつき，暗黙裡に 4 種類エンティティ・サービスを生成する．これらは，データベースの CRUD（Create，Refer，Update，Delete）処理に対応するが，REST の場合は，HTTP の操作（POST，GET，PUT，DELETE）と対応付けを行う．提案手法では，利便性のため参照の処理はいくつかのバリエーションを生成する．**表 3** はエンティティに対する DARS の一般形の一部を示している．

5.9. 仮想表サービス（関連に対するサービス）を生成する

GenDARS は隣接行列を走査し，セルの内容に応じて仮想表の定義（Create View 文）と仮想表サービスの定義を生成する．仮想表を生成する目的は，オブジェクトモデルとリレーショナルモデルのギャップを埋めることと，派生属性値の計算等の小さなビジネスロジックを持たせたいがためである．

提案手法では仮想表サービスは安全のため照会専用サービスと位置付ける．そのため，インスタンスの追加, 変更, 削除に関するサービスはエンティティサービスに委譲する．**表 4** に仮想表に対する DARS の一般形を示す．

4. まとめ

筆者らは旅行会社の販売支援システムの案件に対して提案手法を適用した．13 のエンティティに対して 103 のエンティティ・サービスおよび，13 の関連に対して 77 の仮想表サービス，合計 180 の DARS 群が生成された．結果，実装設計者は実装すべき業務に応じて順番に DARS を呼び出すクライアントを記述していくことが可能となった．適用業務を実装していく際に Swagger UI が強力な利用者ガイドの役割を果たした．設計ドキュメントと実装コードの乖離がないことは生産性に寄与する．

また，提案手法によって，適用業務が要求する表現とドメインモデル（存在従属グラフ）の間に 1 層のレイヤを設けることが容易に実現できるようになった．これにより，ドメインモデルレベルでのカプセル化が可能となり，ドメインモデルに影響する仕様変更を DARS のレイヤである程度吸収することができた．これは RDBMS が提供する仮想表の効果が大きい．仮想表の利用については[10]などのデータベースの教科書でも説明されるが，開発プロセスの早期段階で存在従属グラフと仮想表サービスを積極的に活用する提案は筆者らのオリジナルである．

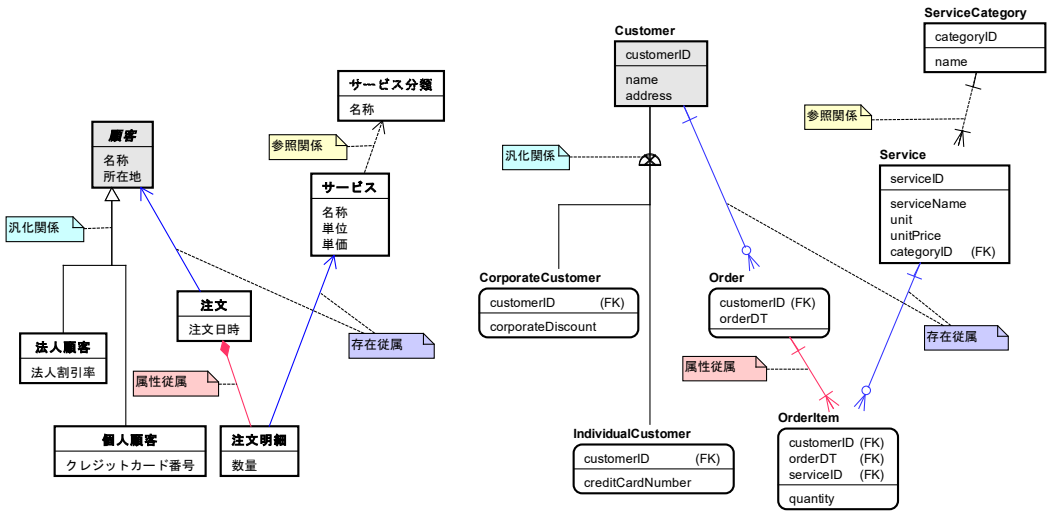

図 2：エンティティの存在従属グラフ（右側の ER 図は識別子の関係を明示するために記載）
存在従属グラフの誘導可能性は HATEOAS の実現に利用

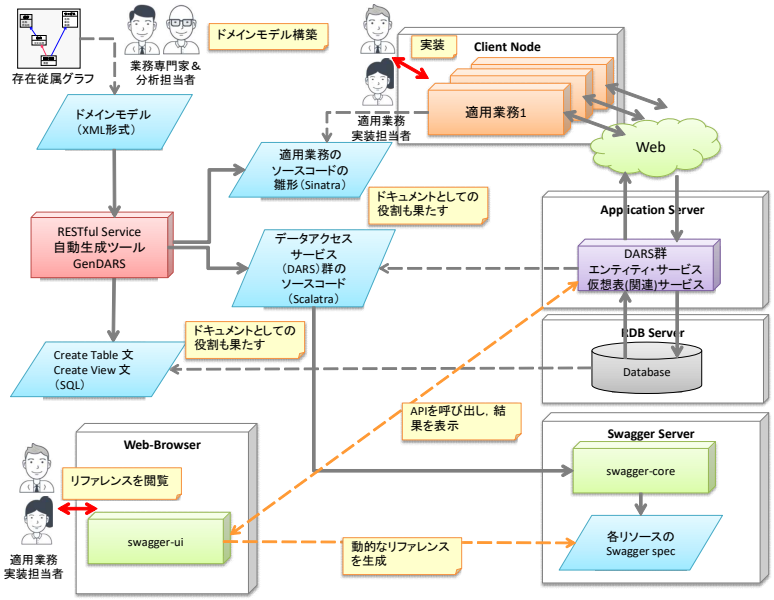

図 3：ツールの連携様式

表 2：存在従属グラフからの REST サービスの検討

(1) 顧客が自分自身を個人顧客として登録する場合
①顧客は，適用業務に氏名，住所，クレジットカード番号などの属性値を提示して，自分を個人顧客として登録することを依頼する．
②適用業務は，その処理の中でサービスPOST Customerを呼び出して，顧客インスタンスを新規登録する．入力された属性値群はリクエスト・ボディに格納されて渡される．新規登録した結果，新規のcustomerIDが採番されるため，それを顧客に示す．
(2) 顧客がサービスを注文する場合
①顧客は顧客登録時に示されたcustomerIDを適用業務に入力する．
②適用業務はGET Customer／{customerID}を呼び出して，顧客の属性値を取得する．そして，それらの属性値に問題がなければ，PUT Order／{customerID}＋{orderDT}を呼び出して，新規注文（見出し）を生成する．ここで，orderDTのDTはDateTimeの略で注文時点のタイムスタンプ値であるとする．
③顧客は注文可能なサービスの一覧を適用業務に要求する．
④適用業務はGET Serviceを呼び出して，提供可能なサービスの一覧を取得し，顧客に示す．
⑤顧客は利用したいサービスをいくつか選択して適用業務にそれらの注文を依頼する
⑥適用業務は選択されたサービスの数だけ，PUT OrderItem／{customerID}＋{orderDT}＋{serviceID}を呼び出して，新規の注文に属性従属させる注文明細を生成する．
(3) 顧客が注文を変更する場合
①顧客は，適用業務 customerIDを入力して，自分が発した注文を照会する．
②適用業務は，GET OrderCutomer／{customerID}を呼び出して，当該顧客の注文一覧を取得し，顧客に示す．なお，OrderCustomerはエンティティではなく関連（仮想表）に対するDARSである．
③顧客は，変更したい注文を1つ選択して適用業務に注文の変更を依頼する．
④適用業務は，GET OrderService／{customerID}＋{orderDT}を呼び出して，注文明細（注文されたサービスの名称と数量および料金）の一覧を取得し，顧客に示す．
⑤顧客はここで，注文明細の内容を変更，または，注文明細の追加，または，注文明細の削除を行うことができる．
(ア) ある注文明細の数量を変更する場合
適用業務は，PUT OrderItem／{customerID}＋{orderDT}＋{serviceID}を呼び出して，当該明細を更新する
(イ) ある注文明細のサービスを変更する場合
適用業務は，DELETE OrderItem／{customerID}＋{orderDT}＋{serviceID}を呼び出して，当該明細を削除するとともに，PUT OrderItem／{customerID}＋{orderDT}＋{serviceID}を呼び出して，変更後のサービスIDの明細を追加する
(ウ) 新規に注文明細を追加する場合
適用業務はPUT OrderItem／{customerID}＋{orderDT}＋{serviceID}を呼び出して，変更後のサービスの明細を追加する
(エ) ある注文明細を削除する場合
適用業務は，DELETE OrderItem／{customerID}＋{orderDT}＋{serviceID}を呼び出して，当該明細を削除する
(4) 顧客が注文をキャンセルする場合
①顧客は，適用業務にcustomerIDを入力して，自分が発した注文を照会する．
②適用業務は，GET OrderCutomer／{customerID}を呼び出して，当該顧客の注文一覧を取得し，顧客に示す．なお，OrderCustomerはエンティティではなく関連（仮想表）についてのDARSである．
③顧客は，キャンセルしたい注文を1つ選択して適用業務に注文のキャンセルを依頼する．
④適用業務は，DELETE Order／{customerID}＋{orderDT}を呼び出して，指定された注文を削除する．
(5) 顧客が自分自身の顧客登録を抹消する場合
①顧客は，適用業務にcustomerIDを入力して，自分の顧客登録の抹消を依頼する．
②適用業務は，DELETE Customer／{customerID}を呼び出して，顧客登録の抹消を結果を提示する．ただし，注文を持っている顧客はその注文の決済が完了するまで，自分の顧客登録を抹消することはできない．

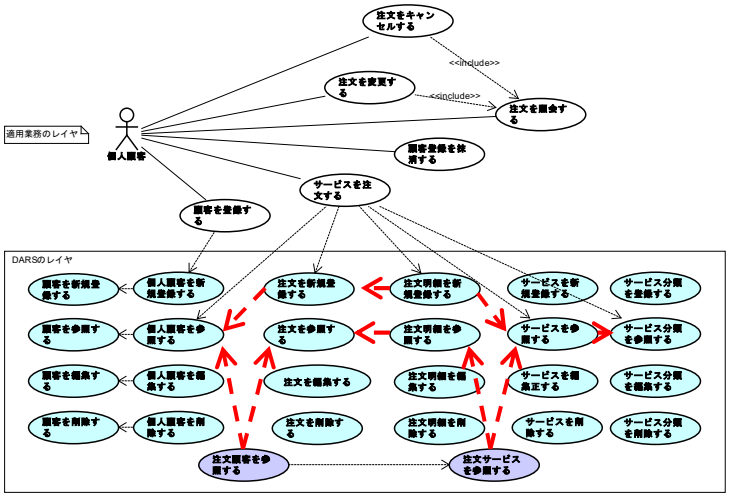

図 4：適用業務のユースケースと DARS の関係（個人顧客の場合．法人顧客も同様）
太い破線は HATEOAS の用いる依存を示す．存在従属グラフの誘導可能性に一致

表 3：エンティティ・サービスの一般形

操作対象	サービスの機能	キー値の採番	HTTP操作	Pathパターン	リクエストのボディ	レスポンスのボディ	レスポンス・ヘッダのステータス	備考
独立エンティティティ	インスタンスの新規登録	サーバ側で連番を自動採番	POST	エンティティ名	新規登録されるインスタンスの属性と属性値の表現	新規登録されたインスタンスの属性と属性値の表現	201 Created 415 Unsupported Media Type 404 Not Found	この操作は安全でもべき等でもない
	特定のインスタンスの照会	なし	GET	エンティティ名/{id}	空	該当したインスタンスの属性と属性値の表現	200 OK 204 No Content 404 Not Found	照会の操作はすべて安全かつべき等である 存在しない{id}が指定された場合はエラーとなる
	特定のインスタンスの編集	なし	PUT	エンティティ名/{id}	編集するインスタンスの属性と属性値の表現ただし，キー属性とその値はなくてもよいし，あっても無視される	編集されたインスタンスの属性と属性値の表現ただし，キー属性とその値はなくてもよいし，あっても無視される	200 OK 204 No Content 415 Unsupported Media Type 404 Not Found	編集の操作はすべて安全ではないが，べき等である 存在しない{id}が指定された場合はエラーとなる
	特定のインスタンスの削除	なし	DELETE	エンティティ名/{id}	空	削除されたインスタンスの属性と属性値の表現	204 No Content 500 Internal Server Error 404 Not Found	削除の操作はすべて安全ではないが，べき等である 存在しない{id}が指定された場合，または，そのインスタンスを参照しているオブジェクトが存在する場合はエラーとなる
独立エンティティティ・従属エンティティ共通	指定範囲のインスタンスの照会	なし	GET	エンティティ名/{from}/{to}	空	該当したすべてのインスタンスの属性と属性値の表現の一覧	200 OK 404 Not Found	最初のインスタンスの位置を0番目とし，fromで指定された位置のインスタンスからtoで指定された位置のインスタンスまでが含まれる
	すべてのインスタンスの照会	なし	GET	エンティティ名/	空	該当したすべてのインスタンスの属性と属性値の表現の一覧	200 OK 404 Not Found	照会の操作はすべて安全かつべき等である
	インスタンスの件数照会	なし	GET	エンティティ名/count	空	件数	200 OK 404 Not Found	件数を照会する操作はすべて安全かつべき等である
	すべてのインスタンスの削除	なし	DELETE	エンティティ名/				あえて実装しない
従属エンティティ	インスタンスの新規登録	クライアント側で既知のためクライアント側が指定	PUT	エンティティ名 /{id1}+{id2}+{id3}+…+{idn}	新規登録されるインスタンスの属性値の表現	新規登録されたインスタンスの属性と属性値の表現	201 Created 415 Unsupported Media Type 404 Not Found	この操作は安全ではないがべき等である
	特定のインスタンスの照会	なし	GET	エンティティ名 /{id1}+{id2}+{id3}+…+{idn}	空	該当したすべてのインスタンスの属性と属性値の表現の一覧	200 OK 204 No Content 404 Not Found	照会の操作はすべて安全かつべき等である
	特定の親に従属するインスタンスの照会	なし	GET	エンティティ名 /{id1}+{id2}+{id3}+…+{id(n−1)}	空	該当したすべてのインスタンスの属性と属性値の表現の一覧	200 OK 204 No Content 404 Not Found	複合識別子の属性がn個の場合，識別子は最低1個からn−1個までを+記号で連結して指定できる．途中の識別子の省略はできない
	指定範囲のインスタンスの照会	なし	GET	エンティティ名/{from}/{to}	空	該当したすべてのインスタンスの属性と属性値の表現の一覧	200 OK 404 Not Found	先頭を0番目としてfrom番目からto番目のインスタンスの一覧を照会

表 4：仮想表（関連）サービスの一般形

操作対象	サービスの機能	キー値の採番	HTTP操作	Pathパターン	リクエストのボディ	レスポンスのボディ	レスポンスのヘッダ	備考
仮想表(ビュー)	特定のインスタンスの照会	なし	GET	仮想表名/{id1}+{id2}+{id3}+…+{idn}	なし	該当したインスタンスの属性と属性値の表現	200 OK 204 No Content 404 Not Found	仮想表の名称は隣接行列のエンティティ名をfrom，toの順に結合したもの
	特定の親に従属するインスタンスの照会	なし	GET	仮想表名/{id1}+{id2}+{id3}+…+{id(n−m)}	なし	該当したすべてのインスタンスの属性と属性値の表現の一覧	200 OK 204 No Content 404 Not Found	複合識別子の属性がn個の場合，識別子は最低1個からn−1個までを+記号で連結して指定できる．途中の識別子の省略はできない
	指定範囲のインスタンスの照会	なし	GET	仮想表名/{from}/{to}	なし	該当したすべてのインスタンスの属性と属性値の表現の一覧	200 OK 404 Not Found	最初のインスタンスの位置を0番目とし，fromで指定された位置のインスタンスからtoで指定された位置のインスタンスまでが含まれる
	すべてのインスタンスの照会	なし	GET	仮想表名/	なし	該当したすべてのインスタンスの属性と属性値の表現の一覧	200 OK 404 Not Found	照会の操作はすべて安全かつべき等である
	インスタンスの件数照会	なし	GET	仮想表名/count	なし	件数	200 OK 404 Not Found	照会の操作はすべて安全かつべき等である

参考文献

[1] Sam Newman：マイクロサービスアーキテクチャ，オライリージャパン（2016 / 2）

[2] R. T. Fielding：Architectural Styles and the Design of Network-based Software Architectures，Irvine， CA（2000 / 8）

[3] S. Ambler，M. Lines：ディシプリンド・アジャイル・デリバリー - エンタープライズ・アジャイル実践ガイド， 翔泳社（2013 / 6）

[4] Philippe Kruchten：ラショナル統一プロセス入門 第 3 版，アスキー（2004 / 11）

[5] L. Richardson，S. Ruby：RESTful Web サービス，オライリージャパン（2007 / 9）

[6] B. Burke：Java による RESTful システム構築，オライリージャパン（2010 / 8）

[7] D. Leffingwell：アジャイルソフトウェア要求―チーム，プログラム，企業のためのリーンな要求プラクティス，翔泳社（2014 / 2）

[8] 金田重郎，井田明男，酒井孝真，熊谷聡志：日本語仕様文からの概念モデリングガイドライン－行為文と関数従属性に基づくクラス図の作成，電子情報通信学会論文誌 D，Vol.J98-D，No.7，pp.1068-1082 （2015 / 7）

[9] 井田明男，金田重郎，熊谷聡志，藤本明莉：存在従属性に着目した論理要件ロバストなドメインモデルの作成－ドメインクラス図をユビキタス言語として用いるために，情報処理学会論文誌，56 巻 5 号 pp 1340-1350 （2015 / 5）

[10] C. Date：データベース実践講義－エンジニアのためのリレーショナル理論，オライリージャパン（2006/2）