情報処理学会論文誌　Vol. 52　No. 3　1126–1135 (Mar. 2011)

シングルテナント型システムからの移行に適したマルチテナント型データベースシステムの構成法

黒田貴之[†1]　副島賢司[†1]　島村栄[†1]

本稿では，単一のテナント（利用者組織）による利用を前提としたシングルテナント型の既存システムから複数のテナントをホスティングする形態を持つマルチテナント型のシステムへの移行（マルチテナント化）を，効率的に実現する手法について論ずる．特に，本来マルチテナント型のシステムでは実現が困難であるテナントごとのデータベーススキーマのカスタマイズ機能を，既存システムからの最小限の変更により実現する手法について述べる．従来のマルチテナント型のリレーショナル・データベースの構成法をマルチテナント化に適用した場合，データベースに拡張用のカラムを大量に追加する必要が生じ，アクセス性能が大幅に劣化した．そこで本研究では，各テナントが拡張したスキーマ上のデータをシリアライズして単一のカラムに挿入することで，マルチテナント化に際して追加すべきカラム数を最少で 2 個に圧縮する手法を提案する．さらに本稿では提案システムの設計と実装について説明し，実験を通じて，提案手法ではカスタマイズ機能の実現によるデータベースのアクセス性能の劣化が従来手法を用いた場合の 30%程度に抑えられる効果について示す．

Architecture of Multi-tenant Database System Suite for Transition from Single-tenant Systems

Takayuki Kuroda,[†1] Kenji Soejima[†1]
and Hisashi Shimamura[†1]

In this paper, we describe an efficient method to transform an existing system to obtain Multi-Tenancy. Multi-Tenancy is a feature of system to host more than one organizations and companies (tenants) by a single shared system instance. Particularly, we focus on how to achieve a function to provide customizable database schemas for each tenant with a minimal change of the existing system. When using a traditional multi-tenant database architecture, a large number of extra columns are needed to be added to the current database tables and it brings on a great performance degradation. Thus, we propose a new method to compress such extra columns into only 2 columns by serializing data assigned to the customized schemas of each tenant and reassigning to a single column. Additionally, we show a design and an implementation of the proposed system and present effects of our proposal along with some experiments. Namely, it reduces the performance degradation caused by the schema customizing function to 30% of that with the traditional method.

1. はじめに

近年，クラウドコンピューティングの台頭にともない，マルチテナント型と呼ばれるシステムの実装形態が注目を集めている[1),2)]．マルチテナント型とは，従来単一のテナント（利用者組織）に割り当てていたリソース（サーバマシン，ミドルウェア，アプリケーションなどのハードウェアやソフトウェア，など）を，複数のテナント間で共有させて効率的に利用するためのシステムの実装形態である．ここでテナントとは，いわゆる B2B2C 型のビジネス形態における中央の B にあたる事業者のことである．たとえば，複数のオンラインショップを一括して提供するマルチテナント型のシステムであれば，各ショップのオーナがテナントであり，各テナントがかかえる顧客がエンドユーザとなる．マルチテナント型のシステムは，各テナントに割り当てるリソースの粒度，すなわちテナントを隔離する単位により様々な類型に分類できる．**図 1** にマルチテナント型のシステムアーキテクチャの類型を示す．図 1 中で左に示される類型ほど，テナントを隔離するレイヤが低く，隔離レベルが高いといえる．一方，右に示される類型ほど，割り当てるリソースの粒度が細かく，その適切な共有と隔離のために高精度な処理制御が必要となるが，テナント間で共有される部位が大きいために高いテナント収容効率が期待できる[1)]．本研究では，図中 (e) に示されるような，単一のアプリケーションで複数のテナントを収容する形態を持つマルチテナント型のシステムを対象とし，これを単にマルチテナント型システムと呼ぶ．

マルチテナント型システムにおいては基本的にはすべてのテナントに同一の機能が提供されるが，テナントごとの個別の機能要件が往々にして発生するため，これに対応するカスタマイズ機能の充実がシステムの利用性向上のためには重要である．文献 2) では，システムが持つカスタマイズ可能な性質を多態性と呼び，それを独自の開発言語や特殊なデータベーススキーマなどを用いて実現している．しかしながら，こうした特殊な実装方式の採用は，開発者に新たな開発手法の習得や適用可能性の検証作業などを強いるものであり，とり

†1 日本電気株式会社
NEC Corporation

わけ既存のシステム資産を活用した効率的な開発は困難であるという課題がある．

そこで筆者らの研究グループでは，既存システムからの移行を前提とし，既存システムの変更を最小限に抑えつつマルチテナント化および多態性の付与を実現する，マルチテナント型システム構成法に関する研究活動を推進してきた．具体的には，システムのマルチテナント化に必要な要素を集約したマルチテナント実現モデルである TIP-Through モデルを考案し，実用化へ向けた取り組みを進めている．

本稿では，システムのマルチテナント化の中でも特にデータベース（DB）のマルチテナント化に着目し，既存システムの DB を基に，その変更を最小限に抑えながら，マルチテナント化および多態性の付与を実現する手法を提案し，その実現性と有効性について述べる．具体的には，各テナントが個別に定義する拡張データをシリアライズして，DB 上の単一のカラムに格納することにより，最少で 2 個のカラム追加によってマルチテナント化と多態性の付与を実現する手法を提案する．本稿でマルチテナント化の対象とする既存システムは，現状広く普及している構成を想定し，リレーショナル・データベース（RDB）と O/R マッピングフレームワークを用いているものとする．そして，このような構成のシステムにおいては，提案手法では従来手法と比較して，カスタマイズ機能の利用による DB のアクセス性能の劣化が 30%程度に抑えられることを示す．

以降，本稿では 2 章で従来のマルチテナント型データベースシステムについて説明し，3 章で提案する選択的カラム圧縮手法ついて述べる．4 章では，まず提案するマルチテナント型データベースシステムの構成を示し，提案システムにおけるデータアクセスの手順，および TIP-Through モデルに基づいたプロトタイプシステムの実装について述べる．続いて 5 章では，プロトタイプシステムを用いた実験を通じて提案システムの効果を示し，その有効性に関する考察を行う．最後に 6 章でまとめと今後の課題について述べる．

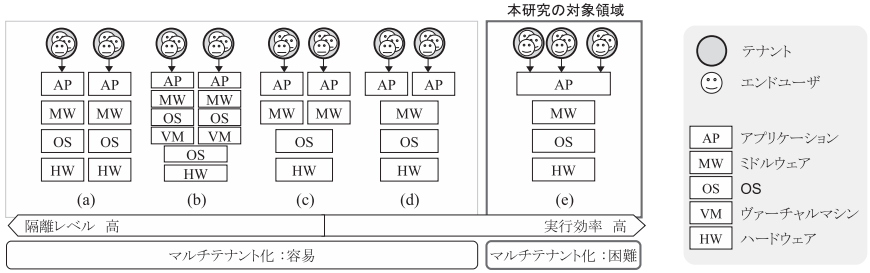

図 1　マルチテナント型のシステムアーキテクチャの類型
Fig. 1　Types of multi-tenant system architectures.

2. 従来のマルチテナント型データベースシステム

2.1 従来のマルチテナント型データベースシステムの概要

従来のカスタマイズ可能なマルチテナント型データベースシステムの構成法には，テナントごとに異なるすべての DB スキーマを単一の抽象的な DB スキーマ内に収容する手法[3]がある．**図 2** に文献 3) における従来のマルチテナント型の DB スキーマの概要を示す．説明の簡便のため，以下では各テナントが独自に拡張した DB スキーマを論理スキーマと呼び，DB 上に定義されている実際の DB スキーマを物理スキーマと呼ぶ．また論理スキーマ上に定義されるカラムを論理カラム，物理スキーマ上に用意されるカラムを物理カラムと呼ぶ．

図 2 は RDB における 1 枚のテーブルを示しており，ここにすべてのテナントのデータが保存される．物理スキーマは，テナント ID，標準カラムおよびカスタムカラムの 3 種類の物理カラムから構成される．テナント ID は各レコードのデータを所有するテナントを示す識別子であり，標準カラムはすべてのテナントにおいて共通に利用されるカラムである．カスタムカラムはテナントごとの独自のスキーマ定義部分を格納するカラムであり，DB スキーマの多態性を実現する要素である．

カスタムカラムの実体は一般的な可変長文字列型のカラムである．テナントが各々の論理スキーマ内に格納するデータは，すべて文字列に変換されて，物理スキーマ上のいずれかのカスタムカラムに挿入される．データの参照時には，逆に文字列から数値型や日付型などの元のデータ型のデータに変換される．各テナントの論理スキーマの定義は別途メタデータ

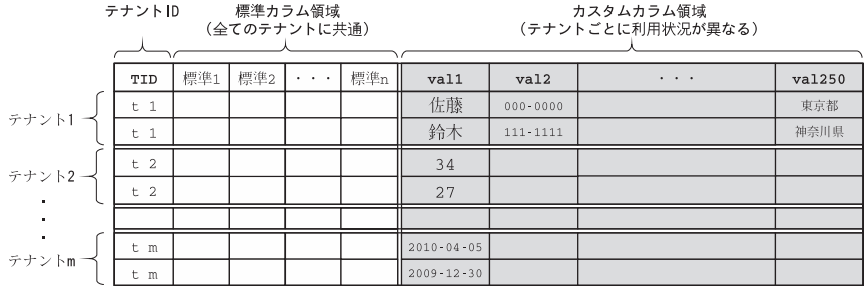

図 2　従来のマルチテナント型の DB スキーマの概要
Fig. 2　An outline of the traditional multi-tenant DB schema.

として管理されており，これに基づいてテナントごとの論理カラムを物理カラムに適切に結び付ける．これにより，テナントごとに個別のテーブルを割り当てているのと同等の利用環境を仮想的に実現している．図 2 には，複数のテナント（1，2，および m）が同一のカスタムカラム（val1）にそれぞれ異なる意味のデータを格納している様子を例示している．また，この例ではテナント 1 は用意された 250 個のカスタムカラムをすべて利用しているのに対し，テナント 2 とテナント m では 1 個ずつしか利用しておらず，残りのカラムは空の状態となっている．このように，利用するカスタムカラムの数はテナントごとに異なる場合がある．

本節で説明した従来手法においては，すべてのテナントの論理スキーマは単一の物理スキーマ内に収容され，各テナントの論理カラムはカスタムカラムとして用意された物理カラムと 1 対 1 で割り当てられる．そのため，すべてのテナントの論理カラムの定義要求を受け入れるには，最も多数の論理カラムの定義を要求するテナントに合わせて，その割当てに必要な数の物理カラムを用意する必要がある．サービスの運用時の逐次的なカラム追加は，それにともなう処理負荷やテーブルロックがサービスへ及ぼす影響が大きいことから，実際には事前に大量の物理カラムが用意され，そのほとんどは空の状態で運用されることとなる．このような理由から，この手法はスパースカラム手法（疎なカラム群による手法）[4] と呼ばれる．

2.2　従来手法の問題点

前節で述べたスパースカラム手法を，本研究の目的である既存システムのマルチテナント化に適用する場合，既存システムの DB スキーマに対して大量のカスタムカラムを追加することとなる．しかしながら，これには下記に示すような様々な問題がある．

- **DB へのアクセス性能の劣化**　物理カラムが大量に存在することにより，それが利用しないカラムであっても，DB へのアクセス性能が劣化する．特に，現在普及している O/R マッピングフレームワークにはカラム数が性能に大きく影響するものがあるため，既存システムがこのような O/R マッピングフレームワークを利用している場合には，カラム数の増大が性能へ与える影響は大きい．
- **適用可能な DB が限定的**　現在普及している一部の DB には定義できるカラム数に制限があるため，既存システムがこのような DB を利用している場合には適用できない．
- **運用管理作業への影響**　既存のカラムが，追加される大量のカラムに紛れ込み，手動での DB スキーマのメンテナンスが困難となる．

特に，1 つ目の問題であげた O/R マッピングフレームワークは，システムの実装と密に結合するものであってマルチテナント化に際して取り除くことは現実的ではないことや，非常に多くの既存システムで利用されていることから，これを利用したシステムにおける性能劣化の解消は重要な課題である．

3.　選択的カラム圧縮手法に基づくマルチテナント型データベースシステムの提案

3.1　選択的カラム圧縮手法の概要

2 章で述べた課題を解決するには，既存システムのマルチテナント化と多態性の導入を，DB スキーマに追加するカラムを最小限に抑えながら達成する必要がある．そこで本研究では，各テナントが定義する論理カラムのうち，個別の物理カラムに割り当てる必要のないものについては，一括して単一の物理カラムに割り当てることにより，従来手法において物理スキーマに定義していた大量のカスタムカラム群を単一のカラムに圧縮する，選択的カラム圧縮手法を提案する．

本提案手法では，各テナントの利用者が登録するデータはシリアライズ処理により単一のデータに変換して DB に格納する．一般にシリアライズとは複数のデータ要素からなる構造化されたデータを単一のデータに可逆変換する処理のことであり，元の構造化されたデータへ戻す処理はデシリアライズと呼ばれる．たとえば，name カラムと age カラムからなる論理スキーマを定義し，name カラムには “suzuki” を格納し age カラムには “20” を格納する場合であれば “name=suzuki&age=20” などの文字列へ変換する処理がシリアライズであり，この文字列から元の “suzuki” と “20” の各データを name カラムと age カラムとの対応が維持された形式で抽出する処理がデシリアライズである．シリアライズにより，複数の論理カラムに割り当てたデータを単一の物理カラムに格納することで，物理カラムを 1 つ用意するだけで任意の数の論理カラムからなる論理スキーマを収容することができるようになる．しかしながら，単純にすべてのデータをシリアライズして DB に保存した場合には，個々のデータをキーにした検索が実行できなくなるほか，DB が持つ制約や演算などの様々な機能も適用できなくなり，DB の利便性が大きく損なわれるという問題がある．そこで本手法では，これらの DB の機能を有効にするため，必要に応じてシリアライズせずに個別のカラムに割り当てる論理カラムを指定できるようにする．

本手法によれば，マルチテナント化のために最低限追加すべきカラムは，テナント ID を格納するカラムと各テナントの論理スキーマを圧縮して格納するカラムの 2 つのみとなる．DB の機能の有効化のために用意するカラムが別途必要であるが，大幅なカラム数の削減が

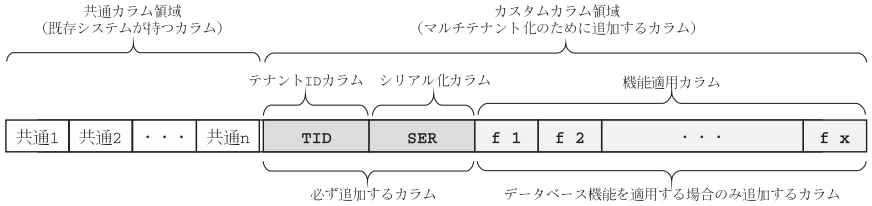

図 3　提案するマルチテナント型データベースシステムの物理スキーマ構成
Fig. 3　The proposed multi-tenant database schema.

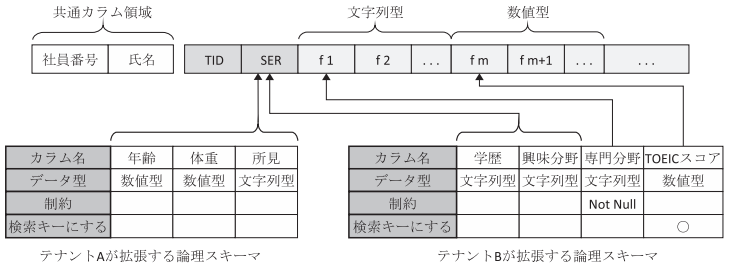

図 4　論理スキーマと物理スキーマとの間のカラムのマッピング例
Fig. 4　An example of column mapping between logical schemas and a physical schema.

可能となり，最小限のカラム追加でマルチテナント化と多態性の導入が実現できる．

以降，本章では 3.2 節で，提案する選択的カラム圧縮手法における物理スキーマの構成について説明し，3.3 節で各テナントによる論理スキーマの定義方法について述べる．

3.2　物理スキーマ構成

図 3 に本手法を用いた，マルチテナント型データベースシステムの物理スキーマ構成を示す．定義されるカラムの領域は共通カラム領域とカスタムカラム領域の 2 つに大別される．このうち共通カラム領域のカラムは，マルチテナント化する対象のシステムが元々備えていた DB スキーマ上のカラムであり，これはマルチテナント化後もすべてのテナントに共通に利用される．一方のカスタムカラム領域のカラムは，マルチテナント化に際して新たに追加されるカラムである．カスタムカラム領域は，テナント ID カラムとシリアル化カラムおよび複数の機能適用カラムから構成される．テナント ID カラムはテナント ID を格納するためのカラムであり，テナント ID とはこのテーブルの各行がどのテナントのデータを格納しているかを示す識別子である．シリアル化カラムは，DB の機能を適用する必要がないデータをまとめてシリアライズしたデータが格納されるカラムである．また機能適用カラムは，検索条件への指定や制約の付加などの DB の機能を適用するデータを個別に格納するカラムである．

カスタムカラム領域内のカラムのうち，テナント ID カラムは DB をマルチテナント化するうえで必要不可欠なカラムであり，シリアル化カラムは DB のカスタマイズ機能を実現し多態性を導入するために最低限必要なカラムである．そのため，本研究で提案するマルチテナント化手法においては，これら 2 つのカラムは必ず追加する必要がある．一方の機能適用カラムについては，各テナントに対して，DB の機能の適用が可能な拡張カラムを最大で何個まで定義可能とするかによって，任意の数だけ追加することとなる．機能適用カラムの数が多いほど各テナントの自由度は向上するが，多すぎると既存手法と同様の問題が再発することになるため適度な数で定義することが重要である．機能適用カラムの適切な定義数については 5 章において述べる．

3.3　論理スキーマの定義方法

テナントごとの論理スキーマの定義方法は，基本的には一般的な DB スキーマの定義方法と同様に，カラム名・データ型・制約の 3 つの情報からなるカラム定義を，必要なカラムの数だけ列挙するものであるが，各カラムに対して “検索キーにするか” の情報を付加できる点が異なる．提案手法では，各論理カラムに何らかの制約の定義があるか，あるいは検索キーにするとの指定があるかを確認し，いずれかの条件にあてはまる論理カラムについては DB の機能が適用されるように個別の機能適用カラムに割り当て，それ以外の論理カラムはすべてシリアル化カラムに割り当てる．こうして作成された，論理カラムと物理カラムの間の割当てに関する情報をカラムマッピング情報と呼ぶ．

図 4 には，本手法における各テナントの論理スキーマと物理スキーマとの間のカラムのマッピング例を示す．ここでは社員情報を管理するテーブルに対して，2 つのテナント A と B が，それぞれ独自の論理スキーマによって拡張カラムを定義している様子を示している．テナント A は健康管理に関連するデータを格納するため，年齢，体重，所見の論理カラムを定義し，テナント B は担当業務の適性の管理に関連するデータを格納するため，学歴，興味分野，専門分野，TOEIC スコアの論理カラムを定義している．テナント A が追加した 3 つの論理カラムはいずれも検索キーなどにする予定がないため，これらはすべてシリアル化カラムに割り当てられている．一方，テナント B が追加した論理カラムでは，学歴と興味分野については，検索キーにする予定はないためシリアル化カラムに割り当てられているが，専門分野については入力を必須とするために Not Null 制約が定義されており，また

TOEIC スコアについてはある得点以上の社員を抽出する操作の実行を予定していることから検索キーにすると指定されているため，これらの 2 つの論理カラムは機能適用カラムに割り当てられている．このとき，機能適用カラムは，論理カラムに指定されたデータ型と機能適用カラムのデータ型を確認したうえで，適合するものを順次割り当てていく．

4. マルチテナント型データベースシステムの設計

4.1 全体構成

本稿で提案するマルチテナント型データベースシステムは，主にテナント管理機能とスキーマ変換機能の 2 つの機能およびデータベースの実体からなる．**図 5** に，提案するマルチテナント型データベースシステムの全体構成を示す．テナント管理機能は，テナント管理者からのテナント自体の追加や削除の要求および論理スキーマの定義に関する設定の要求を受け付け，スキーマ変換機能が動作するために必要な環境を整える機能である．またスキーマ変換機能は，エンドユーザからのデータの登録や参照の要求を受け付け，テナントごとの論理スキーマに基づくデータと物理スキーマ上のデータとの相互変換を行う機能であり，エンドユーザに対してマルチテナント化する前の既存システムと同様のインタフェースを提供するとともに，仮想的に専用の DB 利用環境を提供する機能を持つ．

4.2 テナント管理機能の詳細

テナント管理機能は，テナント依存情報およびテナント依存情報作成機能からなる．ここで，テナント依存情報とは以下の 3 つの情報をテナントごとにまとめたものである．

- **テナント ID**　テナントごとに割り当てられる識別子．

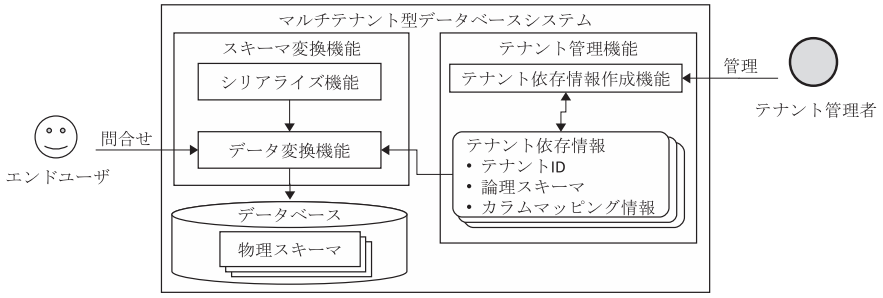

図 5　提案するマルチテナント型データベースシステムの全体構成
Fig. 5　The whole architecture of the proposed multi-tenant database system.

- **論理スキーマ**　テナントごとの拡張カラムに関する論理的なスキーマの定義．具体的には，カラム名，データ型，制約および検索キーにするかの 4 つの情報から構成される論理カラムの定義を列挙したもの．
- **カラムマッピング情報**　当該テナント依存情報内の論理スキーマに定義された各論理カラムと物理カラムとの対応関係を示す情報．

テナント管理機能は，テナント管理者から論理スキーマの定義を受け取ると，各論理カラムの定義情報を参照し，3.3 節で述べた手順に従ってカラムマッピング情報を作成する．そして，テナント ID および論理スキーマの定義とともに，テナント依存情報として保存する．また，テナント管理者からテナントを削除する要求を受け取った場合には，当該テナントのテナント ID を持つテナント依存情報を削除するとともに，DB から対応するテナント ID を持つレコードを削除する．

4.3 スキーマ変換機能の詳細

スキーマ変換機能は，主にデータ変換機能とシリアライズ機能の 2 つの機能を持つ．データ変換機能は，エンドユーザからデータの登録や参照の要求を受け付けると，まずリクエストの対象となるテナントを識別し，当該テナントに関連したカラムマッピング情報を取得して，これに基づいて各データに対して論理スキーマと物理スキーマの間のカラム割当ての変換処理を実施する．この際，シリアル化カラムに割り当てられたデータはシリアライズ機能に委譲し，ここでシリアライズおよびデシリアライズを実行する．リクエストの対象となるテナントの識別処理については，本研究を通じて開発を進めてきた TIP-Through モデルに基づくフレームワークの機能を用いて実現する．TIP-Through モデルについては次節で概要を述べる．

スキーマ変換機能による，データの登録と参照の処理フローは以下のとおりである．

データ登録の処理フロー

S1　エンドユーザからデータの登録要求を受け付け，TIP-Through の機能を用いてリクエストの対象となるテナントを識別し，これに関連するテナント依存情報を取得する．

S2　取得したテナント依存情報内のカラムマッピング情報を参照し，論理スキーマ内にシリアル化カラムに割り当てられた論理カラムがあるかを確認する．ある場合には S3 へ進み，ない場合には S4 へ進む．

S3　シリアル化カラムに割り当てられた論理カラム内のデータを，すべてまとめてシリアライズし，S4 へ進む．

S4　テナント ID カラムに取得したテナント依存情報内のテナント ID を格納する．S3 で

シリアライズされたデータがある場合には，それをシリアル化カラムへ格納する．機能適用カラムへ割り当てられた論理カラムがある場合には，それらの論理カラム内のデータを対応する各機能適用カラムへ格納する．

データ参照の処理フロー

S1　エンドユーザからデータの参照要求を受け付け，TIP-Through の機能を用いてリクエストの対象となるテナントを識別し，これに関連するテナント依存情報を取得する．

S2　取得したテナント依存情報内のカラムマッピング情報を参照し，ユーザから受け付けた参照要求のクエリ内の論理カラム名を対応する物理カラム名へ変換するとともに，クエリの条件にテナント ID を加えて，DB に問い合わせる．

S3　カラムマッピング情報を参照し，論理スキーマ内にシリアル化カラムに割り当てられた論理カラムがあるかを確認する．ある場合には S4 へ進み，ない場合には S5 へ進む．

S4　DB から取得したシリアル化カラム内のデータをデシリアライズし，S5 へ進む．

S5　カラムマッピング情報を参照し，DB から取得したデータを対応する各論理カラムに割り当てる．

4.4　TIP-Through モデルの概要

前節で述べた，リクエストの対象となるテナントの識別処理は，筆者らが研究開発を進めている TIP-Through モデルに基づくフレームワークの機能によって実現する．TIP-Through モデルは，テナントの識別機能，識別情報の伝達機能，処理の割振り機能の 3 つの機能から構成される．テナントの識別機能は，システムの処理開始時に処理を起動したリクエストの対象となるテナントを識別し，処理の ID とテナントの ID を組にして記録する機能である．たとえば “http://www.example.com/tenant1/” へのアクセスに対しては “tenant1” という文字列をテナント ID と識別するなど，アプリケーションごとの識別方法に対応する実装を持つ．識別情報の伝達機能は，テナントの識別機能から受け付けた処理の ID とテナント ID の組に関する情報を保持し，システムの処理フローの各所へ伝達する機能である．また処理の割振り機能は，システムの処理に割り込んで，テナント固有のリソースに対する処理を振り分ける機能である．以上の機能により，システム全体の共用リソースを複製・分割してテナントに割り当てたり，DB などのテナント固有データをテナントごとに適切に切り替えたりすることができるようになる．

テナントの識別機能や処理の割振り機能は，AOP（Aspect Oriented Programming）の仕組みを用いて，既存システムの各所へ割り込ませる実装形態を想定している．

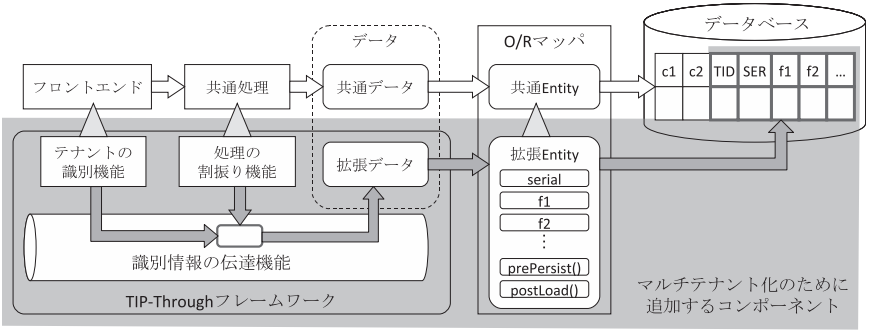

図 6　マルチテナント型データベースシステムの実装
Fig. 6　An implementation of the proposed multi-tenant database system.

4.5　マルチテナント型データベースシステムの実装

本研究の目的は，システムのマルチテナント化に適したマルチテナント型データベースシステムの構成法の確立である．そこで，本稿における提案システムのプロトタイプの実装では，まず一般的な構成を持つシステムを想定し，それに対してマルチテナント化に必要な機能を追加する形式をとる．**図 6** に，本稿におけるマルチテナント型データベースシステム実装の概要を示す．図中の網掛けの部分が，マルチテナント化のために追加される拡張コンポーネントであり，それ以外の部分が既存のシングルテナント型のシステムである．図中の矢印は，データの登録処理の流れを表している．

本稿では，既存システムとして O/R マッピングフレームワークによる DB アクセス機構を備えた Web アプリケーションを想定している．この既存システムは，フロントエンドでユーザからのリクエストを受け付け，共通処理の中で登録されるデータを取得し，それを O/R マッピングフレームワークが管理するエンティティオブジェクトへ格納した後，O/R マッピングフレームワークの機能によって DB へ格納する．この既存システムに対し，AOP の機能を用いてマルチテナント化のための拡張コンポーネントを挿入する．まずフロントエンドに対して前節で述べた TIP-Through フレームワークのテナントの識別機能を挿入し，ここでリクエストの対象となるテナントを識別して，それを識別情報の伝達機能に記録する．また共通処理には，処理の割振り機能によってユーザからの拡張データの入力を受け付ける処理を挿入し，これにより拡張データをテナント識別情報とともに記録する．この拡張データは，テナントごとの論理スキーマに準じた形式を持つデータである．さらに共通エ

ンティティには，拡張エンティティとして，シリアル化カラムと機能適用カラムに対応したフィールド（serial や f1，f2 など）およびそれらのアクセサを追加する．また，論理スキーマと物理スキーマの間の形式の違いを吸収する処理を行うため，prePersist と postLoad の 2 つのメソッドもあわせて挿入する．prePersist はエンティティが DB に記録される直前にコールされるメソッドであり，拡張データを論理スキーマの形式から物理スキーマの形式に変換する処理を行う．postLoad は，逆に DB からデータがロードされた直後にコールされ，拡張データを物理スキーマの形式から論理スキーマの形式に変換する処理を行う．

今回の実装では Java ベースのシステムを想定し，DB アクセス層の実装としては J2EE の標準仕様である JPA 仕様に準拠するものとした．O/R マッピングフレームワークには，最も普及している JPA の実装である Hibernate（3.3.1.GA）[5] を用いた．また AOP の実現のためのライブラリとしては AspectJ（1.6.5）[6] を用い，シリアライズ処理のツールには処理速度を重視して ProtocolBuffer（2.1.0）[7] を用いた．

5. 実験・評価

5.1 予備実験

本研究の有効性を示す実験に先立ち，O/R マッピングフレームワーク（Hibernate）を用いた DB アクセスにおける，カラム数と消費時間の関係を明らかにする予備実験を実施した．実験内容は，カラム数およびデータアクセスに用いられるエンティティオブジェクトのプロパティ数を変化させながらデータの書き込みと読み出しを実施し，各処理の消費時間を計測するものである．この結果を**表 1** に示す．計測対象の処理内容としては，書き込み時間は 1 トランザクションあたり 1 件のデータ登録を 100 回繰り返すのに要した時間とし，読み出し時間は書き込み時間の計測時に登録された 100 件のデータを 1 トランザクションで読み出す操作を 100 回繰り返す処理に要した時間とする．なお，純粋にカラム数による影響を測るため，各カラムのデータはすべて空文字とした．DB は Oracle10g（Express Edition Release 2），PostgreSQL8.3.7，MySQL5.1（MyISAM）の 3 種類を用意し，いずれもローカルマシン上に配置して実験を行った．傾向はいずれの DB もおおむね同様であったので結果については Oracle のみ示す．実験に用いたマシンのスペックは，CPU が Pentium(R)4 CPU 3.40 GHz，メモリが 2.00 GB RAM，HDD の容量が 232 GB，OS が Microsoft Windows XP Professional Version 2002 Service Pack3 である．

表 1　Hibernate を用いた DB アクセスにおけるカラム数と消費時間の関係
Table 1　A relationship between column numbers and response times in DB access with Hibernate.

カラム数	0	40	75	160	250
書き込み時間（ミリ秒）	543.8	762.5	974.8	1,468.8	2,020.3
読み出し時間（ミリ秒）	954.7	1,895.3	2,701.6	4,715.6	6,875.3

この結果から，Hibernate においては，カラム数の増加が DB アクセス性能の大幅な劣化をもたらすことが分かる．

5.2 実験方法

次に，4.5 節で述べたプロトタイプシステムを用いて提案手法の有効性を検証する実験を行う．基本的な実験方法としては，提案する選択的カラム圧縮手法と既存のスパースカラム手法のそれぞれでデータの書き込みと読み出しを実行し，これらの処理にかかる時間を計測する．この操作を物理カラムの数と論理カラムの数を変化させながら繰り返し実行し，カラムの数が性能に与える影響と，手法による影響度合いの違いを明らかにする．

本実験における DB アクセスは，すべて Hibernate を介して行う．共通エンティティには，共通カラムに対応する c1 から c10 までの 10 個のフィールドとそのアクセサを定義する．また本実験では最大で 250 個の論理カラムの定義を許容するものとし，前記共通エンティティに対して，それぞれの手法に応じた次の拡張を行う．すなわち，既存のスパースカラム手法においては，論理カラムに対応する val1 から val250 までの 250 個のフィールドとそのアクセサを追加する．一方の提案手法においては，シリアル化カラムに対応する serial フィールドと，許容する機能適用カラムの個数 F に応じて f1 から fF までの F 個のフィールドおよびそのアクセサを追加する．機能適用カラムの個数 F としては，0 個，40 個，75 個，160 個，250 個の 5 つのパターンを用意する．

実際に扱われるデータ数，すなわち論理カラムの数としては，0 個，40 個，75 個，160 個，250 個の 5 つのパターンを試行する．書き込みフェーズにおいて，スパースカラム手法では論理カラム内のデータを val1 から val250 までの各フィールドにセットする．論理カラムの個数が 250 個より少ない場合には，利用しないフィールドが存在することとなる．また提案手法においては，今回の実験ではすべての論理カラムがシリアル化カラムに割り当てられるものとし，全データをまとめてシリアライズして serial フィールドにセットする．f1 から fF までのフィールドは利用しない．計測対象の処理内容は前節の予備実験と同様に，書き込み時間は 1 トランザクションあたり 1 件のデータ登録を 100 回繰り返すのに要した時間とし，読み出し時間は書き込み時間の計測時に登録された 100 件のデータを 1 トランザクションで読み出す操作を 100 回繰り返す処理に要した時間とする．提案手法においてはシリアライズ処理にあたるデータの集約や展開の処理までを計測対象とする．各カラムに

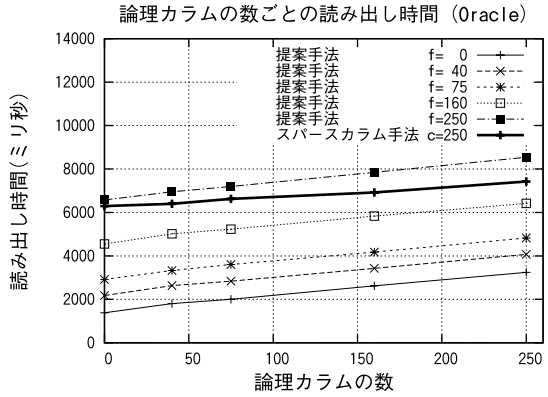

図 7　100 件のデータ読み出し 100 回にかかる時間
Fig. 7　Response time [msec] to read 100 records 100 times.

格納されるデータはすべて 10 文字のランダムな文字列とし，対応する物理カラムの型は最大 255 文字の可変長文字列型とする．シリアル化カラムの型としては Blob（Binary Large Object）型を用いる．また，実験用マシンおよび DB のスペックは前節の予備実験と同様である．

250 個という論理カラムの個数は，2 章で述べた既存研究[3] における実施例を元に設定した．この研究の著者らは現在最も著名なカスタマイズ可能なマルチテナント型サービスの 1 つを運営しており，その実施例はマルチテナント型システムの実験環境の参考値として一定の妥当性を有していると考えられる．また，共通カラムの数や実施するトランザクションの内容などは，マルチテナント化の対象となる既存システムの業務要件に依存する要素であるため，現在の一般的な業務システムとして妥当な状況を想定したものである．

5.3 実 験 結 果

図 7 に Oracle DB における読み出し処理の実験結果のグラフを示す．横軸は論理カラムの数であり，縦軸は各処理に要した時間（ミリ秒）である．図中の太線のグラフは，従来手法において拡張カラムを格納するための物理カラムを 250 個用意した場合の結果であり，それ以外のグラフは提案手法における物理スキーマ上に確保される機能適用カラムの個数に応じた結果である．凡例の f は機能適用カラムの数を示し，c は論理カラムのデータを格納するために確保された物理カラムの数を示している．

この結果から，論理カラムの数および物理カラムの数が多いほど読み込みにかかる時間が大きくなり，性能が劣化することが確認できる．これは論理カラムの数が多いほどデータの転送とスキーマ変換処理（デシリアライズ処理を含む）にかかる時間が増加するためであると考えられる．また予備実験でも確認したとおり，物理カラムの数が多いほど，実際にはデータが転送されていなくても性能が劣化することが確認できる．書き込み処理や他の DB における実験結果の記載は省略したが，おおむね同様の傾向が表れている．

5.4 評　　価

5.3 節で示した実験結果より，提案手法ではマルチテナント化のために既存システムの DB に追加すべきカラム数を大幅に削減できることが確認できた．物理カラム数の削減は O/R マッピングフレームワークを用いた DB アクセスにおける性能改善への寄与が大きく，Hibernate を用いた今回の実装では大幅な性能改善効果が見られた．マルチテナント化する前の性能に対する，250 個の論理カラムを利用した場合の性能劣化の度合いを比較すると，Oracle を用いた場合の読み込み性能については，従来手法では 6 秒程度増加してしまっているのに対し，提案手法ですべての論理カラムをシリアライズして格納する場合には 1.9 秒程度の増加にとどまっており，性能劣化は 31%に抑えられている．最も顕著な改善効果が見られたのは PostgreSQL における読み込み性能であり，性能劣化は 13%程度に抑えられた．

また，今回の実験では定義可能な論理カラムの最大数を 250 個としたが，提案手法においてはより多くの論理カラムを定義することも可能である．MyISAM では 1 つのテーブルに定義可能なカラムに制限があり，すべてのカラムを 255 文字の可変長文字列型とした場合には，共通カラムと合わせて最大で 85 個までしかカラムを定義できない．そのため，従来手法を用いた場合には 250 個もの論理カラムは定義できないが，提案手法においてはシリアル化カラムを利用すれば可能である．また既存の DB スキーマからの変更が小さいことにより，運用管理やメンテナンスなどの作業への影響も少なくなり，これらの作業に利用される周辺システムの修正コストの削減にも寄与すると考えられる．

シリアル化カラム内に格納される個々のデータには，index や列統計などの問合せ最適化に必要な情報が作成されないが，シリアル化カラムに格納されるデータはそもそも検索などには利用されないことを前提としているので，これによる性能への影響は軽微である．

5.5 考　　察

5.5.1 提案手法の有効範囲

本研究の提案手法では，データを検索条件に指定するためには，当該カラムをシリアライズせずに個別の機能適用カラムに格納する必要がある．そのため，検索条件に指定可能なカラム数を増やすには，その分だけ機能適用カラムの数も増やす必要がある．しかしながら，

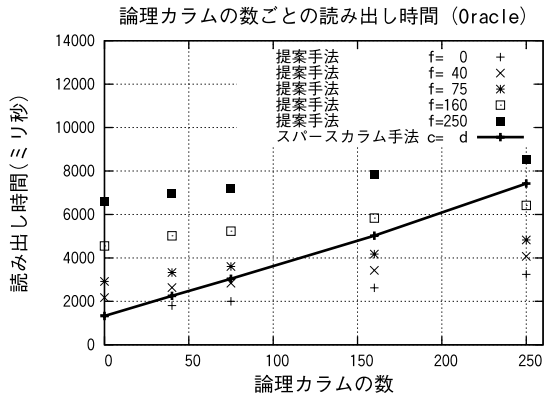

図 8　スパースカラム手法において論理カラム数と物理カラム数を一致させた場合の結果との比較
Fig. 8　Results with the existing scheme when the numbers of physical and logical columns are equal.

本提案手法ではデータをシリアライズしてまとめ，必要な物理カラム数を削減することでDB アクセスの性能を向上させているため，機能適用カラムを増やすほど提案手法の性能改善効果は失われてしまう．ここで**図 8** に，既存のスパースカラム手法において論理カラム数と物理カラム数を一致させた場合の結果との比較のグラフを示す．凡例の f は機能適用カラムの数，c は論理カラムのデータ格納するために確保された物理カラムの数を示し，d は論理カラムの数を示す．

このグラフから，想定される論理カラムと同程度の数の機能適用カラムを用意する場合以外は，おおむね提案手法の方が良い性能が出ていることが分かる．論理カラムが 250 個の場合の性能を比較すると，提案手法で用意する機能適用カラムの数を論理カラムの数と同じ 250 個とした場合よりは，従来手法の方が性能が良いが，機能適用カラムの数を 160 個に抑えた場合には提案手法の方が性能が良くなっている．このように，本手法は論理カラムのほぼすべてが機能適用カラムでない限りは従来手法よりも性能的に優位であり，特に想定される論理カラムの利用数のうち，必要となる機能適用カラムの数が少ないほど優れた性能を発揮するといえる．

5.5.2　他のマルチテナント型データベースシステムの構成法

本稿では，DB の種類として RDB を想定し，提案手法に対する比較対象としてスパースカラム手法を取り上げてきたが，既存のマルチテナント型データベースシステムの構成法には他にもいくつかの手法が存在する[4),8)]．

文献 8) では，既存の構成法として (a) Private table Layout，(b) Extension table Layout，(c) Universal table Layout，(d) Pivot table Layout の 4 種類をあげ，さらに (d) の Pivot table Layout を発展させた独自の構成法 (e) を提案している．これらの手法のうち (a) は各テナントに完全に別個のテーブルを割り当てる方法であり，(b) と (d) および (e) はテナントごとの拡張部分のみ別テーブルに分離する方法である．また (c) は，本稿におけるスパースカラム手法と同じ方法である．各テナントに別個のテーブルを割り当てる方法は，テナントの隔離レベルが高く，個別のカスタマイズやメンテナンスがしやすいという利点はあるが，一方で共有される部分が少ないため全体的な実行効率が低く，共通部分のメンテナンスが困難であるという欠点が指摘されている．また，テナントごとの拡張部分を別テーブルに分離する手法では，DB アクセスの際にはつねにテーブルの結合（join）をともなうこととなり，これが性能へ与える影響は大きい．効率的な分離方法に関する研究はさかんに進められているが，このような複雑なテーブル構造を持つアプローチを既存システムのマルチテナント化に採用する場合，テナントの増加にともなってテーブルの管理コストが増大したり，対象システムの構成が限定されたりするなど，性能面以外の課題も多く残されている．

また，文献 4) では，Key-Value ストアを用いたマルチテナント型 DB システムの構成に関する取り組みが紹介されている．Key-Value ストアやドキュメント DB などのスキーマレスな DB は，高い柔軟性と実行効率をあわせ持つ点においてはマルチテナント型 DB に適していると考えられる．しかしながら，これらの DB にはテーブルの結合やトランザクション制御などの，RDB で提供される多くの機能が実現できないという欠点がある．本研究では既存システムからの移行を前提としているため，現状のシステムで広く利用されている RDB の機能は必須であり，スキーマレスな DB の適用は困難であると考えられる．

以上の理由から，本研究では RDB を想定し，単一のテーブルによるシンプルな構成法を基本とした．しかしながら，複数テーブルを用いた構成法との性能比較や，一部にスキーマレスな DB を用いるなどの取り組みは重要であると考えており，これについては今後の課題とする．

6. おわりに

本稿では，アプリケーションのマルチテナント化に適したマルチテナント型データベースシステムの構成法の確立を目的とし，既存システムが持つ DB スキーマからの変更を最小限に抑えつつテナントごとの DB スキーマのカスタマイズ機能を実現する，選択的カラム

圧縮手法を提案した．本手法では，検索などの DB の機能を適用する必要のないデータについては，シリアライズして単一のカラムに格納することにより，最少で 2 個のカラムの追加によって DB のマルチテナント化とカスタマイズ機能の提供を実現できる．本稿では，プロトタイプシステムを用いた実験を通じて，提案手法の実現性と有効性を確認した．具体的には，提案手法によって DB スキーマに追加すべきカラム数が削減され，これによりカスタマイズ機能を利用した場合の DB のアクセス性能の劣化が，従来手法を用いた場合の 30%程度に抑えられることを示した．

今後は，さらに多くのマルチテナント型データベースシステムの構成法との比較検討を進めるとともに，テナントごとのサービス品質やセキュリティの観点からマルチテナント化技術の実用化を推進していく．

参 考 文 献

1) Guo, J.C., Sun, W., Huang, Y., Wang, H.Z. and Gao, B.: A Framework for Native Multi-Tenancy Application Development and Management, *The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services* (*CEC-EEE 2007*), pp.551–558 (2007).
2) Weissman, D.C. and Bobrowski, S.: The Design of the Force.com Multitenant Internet Application Development Platform, *SIGMOD '09: Proc. 35th SIGMOD International Conference on Management of Data*, ACM, pp.889–896 (2009).
3) 日本特許，ワイズマン クレイグ，セールスフォース ドット コム インコーポレイティッド：マルチテナント・データベース・システムにおけるカスタム・エンティティおよびフィールド，特表 2007-531941(P2007-531941A) (2007.11.8).
4) Aulbach, S., Jacobs, D., Kemper, A. and Seibold, M.: A Comparison of Flexible Schemas for Software as a Service, *SIGMOD '09: Proc. 35th SIGMOD International Conference on Management of Data*, ACM, pp.881–888 (2009).
5) Hibernate – Jboss Community. http://www.hibernate.org/
6) The AspectJ Project. http://www.eclipse.org/aspectj/
7) protobuf – Project Hosting on Google Code. http://code.google.com/p/protobuf/
8) Aulbach, S., Grust, T., Jacobs, D., Kemper, A. and Rittinger, J.: Multi-Tenant Databases for Software as a Service: Schema-Mapping Techniques, *SIGMOD '08: Proc. 2008 ACM SIGMOD International Conference on Management of Data*, ACM, pp.1195–1206 (2008).

(平成 22 年 5 月 20 日受付)
(平成 22 年 12 月 1 日採録)

黒田　貴之（正会員）

2009 年東北大学大学院情報科学研究科情報基礎科学専攻博士課程修了．同年 NEC 入社．大規模サービス向けアプリケーションプラットフォームの研究に従事．情報科学博士．

副島　賢司

1999 年 NEC 入社．大規模金融システム開発，グリッドコンピューティング，情報家電プラットフォーム技術の研究等を経て，現在，大規模サービス向けアプリケーションプラットフォームの研究に従事．

島村　　栄（正会員）

1995 年大阪大学大学院基礎工学研究科情報工学専攻修士課程修了．同年 NEC 入社．Web サービス，SOA 技術の研究および製品化を経て，現在，大規模サービス向けアプリケーションプラットフォームの研究に従事．