Vol.2011-DD-81 No.3
2011/7/21

# Web ベースの表形式フォームシステムでの YAML フォーム構造記述からのフォーム自動生成

近藤浩志[†]　　金子邦彦[††]

九州大学病院の診療経過等報告システムのインシデントレポートのフォームを参考に, Web フォームを構成する要素をフォーム記述データとして YAML 形式で記述した．そのデータから表形式の Web フォームを生成するシステムを作成し，インシデントレポートの内容を再現した．また，params ハッシュテーブルから送信されたフォームデータだけを取り出し，JSON 文字列に変換することで，TEXT 型データとしてデータベースのテーブルに保存した．その際，送信した内容をブラウザに表示し，それとテーブルの内容を比較して正しく保存されているかどうかを見た．

# Automatically Generation of Forms from YAML Form Structural Description on Web-based Tabular Form System

Koji Kondo[†]　and Kunihiko Kaneko[††]

Referring to the form of incident reports in the transitional medical reporting system of Kyushu University Hospital, we described the elements that make up the Web form as "Form Description Data" in YAML. Then, we created a system to generate a tabular Web form from that data, and reproduced contents of the incident report. In addition, form data retrieved from the hash table named params was converted to JSON, and stored in the database table as type TEXT. At that time, displayed contents of submission on the Web browser, and looked at whether it is stored properly by comparing the contents of the table with it.

## 1. はじめに

Web 上でのアンケート調査やレポート提出，ショッピング等において，ユーザーの情報を入力してもらう際にフォームが多く利用されている．九州大学病院の診療経過等報告システムでも，Web 上でフォームを用いてインシデントレポートの提出を行っている（図 1）．インシデントレポートとは，日常診療の現場でひやりとしたりはっとした経験に関する報告書で, 医療事故の発生を未然に防ぐための重要なデータとなる．このインシデントレポートを参考にして，そのフォームを構成する要素をフォーム記述データとして定義し，YAML 形式で構造記述を行った．このとき，インシデントレポートの内容を再現できるように項目の構造は親・子・孫の深さ 3 になるように記述した．本稿では，このようなフォーム記述データから，表形式で表現されたフォームを生成するようなシステムを Ruby と Ruby on Rails を用いて作成した．そして，インシデントレポートのフォーム記述データを用意し，このシステムを通してブラウザにフォームが崩れることなく表示されるかどうかを確認した．また，フォームに入力された情報をフォームデータと呼ぶことにし，インシデントレポートのフォームデータの保存を行った．Rails を利用しているので，表示したインシデントレポートのフォームに適当に入力を行って送信ボタンを押した際に，params というハッシュテーブルからフォームデータだけを取り出すことにした．ハッシュテーブルをそのままの状態で保存するのではなく，JSON という文字列に変換することで TEXT 型データとしてデータベースのテーブルに保存できるようにした．送信した内容をブラウザに表示し，それとテーブルの中身を目視で確認することで正しく保存されているかどうかを見た．

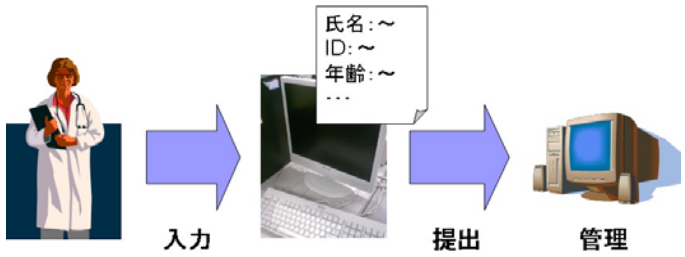


図 1　インシデントレポート提出の流れ

[†] 九州大学 大学院 システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University
[††] 九州大学 大学院 システム情報科学研究院
Graduate School of Information Science and Electrical Engineering, Kyushu University

## 2. 表形式フォームとフォーム記述データのYAML記述

### 2.1 表形式フォーム

このシステムにおいて，生成されるフォームは全て表形式で表現されたものとなる．図 2 に表形式フォームの例を示す．全てのフォームは 1 つ以上の項目で構成されており，全体として 2 列と複数行から成る表形式になっている．1 列目には各項目の番号が表示され，2 列目には各項目の内容が表示されている．各項目は，それぞれ項目名と 1 つ以上のパーツによって出来ているので，項目 1 つは 2 列と 2 行以上で構成される．項目の構造に関しては 2.2 の中で述べる．

ここでパーツと呼んでいるものは，インシデントレポートのフォームを参考に定義したフォームの構成要素（入力部分やボタン）のことを指す．1 行テキスト（文字列）の入力を行ってもらうためのパーツをテキストフィールド，1 行以上のテキスト（文字列）を入力してもらうためのパーツをテキストエリアとする．また，複数の選択肢の中から 1 つだけを選択してもらいたい場合に利用するパーツをラジオボタン，1 つ以上の選択肢の中から任意数選択してもらいたい場合に利用するパーツをチェックボックス，選択肢のリストを用意してその中から 1 つを選択してもらいたい場合に利用するパーツをセレクトメニューと呼ぶ．それ以外に，フォームに入力された情報を送信するためのボタンとなるパーツとしてサブミットボタンがある．

フォームの背景や文字の色，表の枠線の太さやセル内の空白等のレイアウトに関する要素はここでは考えないものとして，フォームのレイアウトは固定とする．

### 2.2 YAML構造記述

YAML 形式で記述されたデータは文字列であるが，そのデータは配列やハッシュに変換することができる．あるフォーム記述データを図 3 のような形で，YAML 形式を用いて記述する．これは，フォーム内の各項目を構成する要素をそれぞれ 1 つのハッシュのリストとして記述しており，フォーム記述データ全体は要素がハッシュのリストである配列で表現される．

ハッシュのリストに含まれるハッシュの種類は，項目内に使用されているパーツによって異なる部分もあるが基本的には同じである．基本の部分はまず，”id”をキーとするハッシュ値があり，これは整数値で項目の番号を示している．”item_name”をキーとする文字列は項目名を表しており，”sub”をキーとする文字列は項目のサブタイトルや項目に関する特記等を表す．また，”required_label”をキーとする真理値は必須項目であるかどうかを示すもので，true の場合は項目名に「(必須)」というラベルを付与する．”child”は項目の構造に関するものであり後述する．”parts”をキーとするシンボルはパーツの種類を示すものであり，”name”をキーとする文字列はフォームデータが送信される際にどの項目のどの入力なのかを識別するためのもので，HTML タグ内の

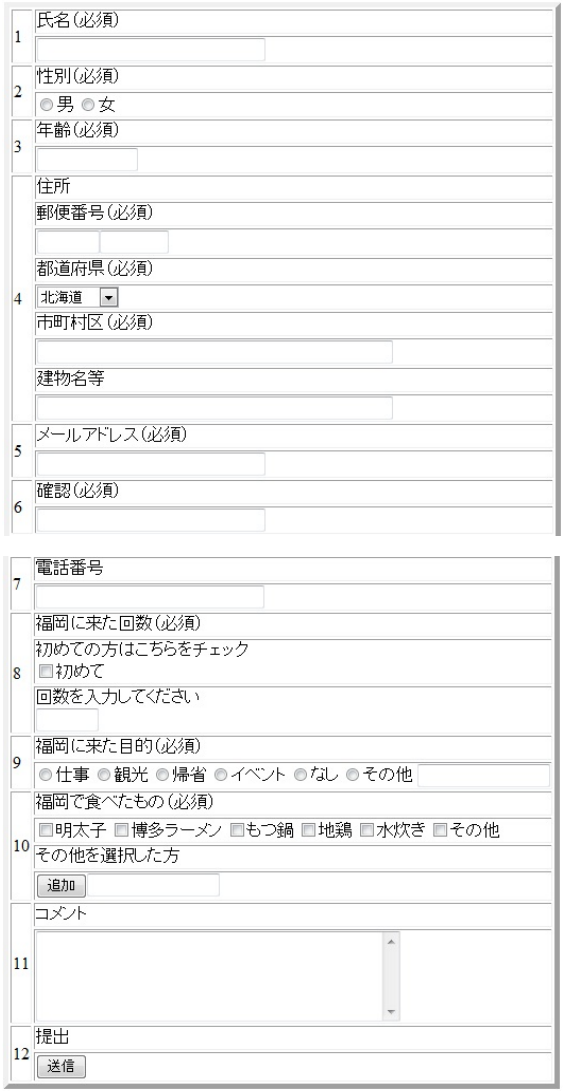


図 2　表形式フォーム例

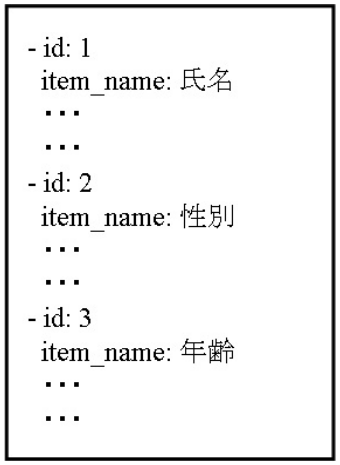


図 3　フォーム記述データ

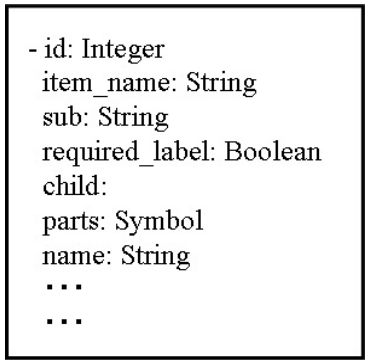


図 4　項目の YAML 記述（基本形）

name 属性の値になる．

続いて項目の構造に関して述べる．項目に関しても，インシデントレポートを参考にしており，そのフォームを再現する上で必要だと考えられる構造を定義している．基本的な項目は項目名と 1 つのパーツによって構成される．この場合，項目の YAML 記述は図 4 のように定義する．”child”キーのハッシュ値は指定しない．”name”キーの記述以降にはパーツ固有のハッシュ値に関する内容が記述される．各項目はその中にさらに複数の項目を持つことがある．例として図 5 のようなものが挙げられる．これは「住所」という項目の中に，郵便番号や都道府県といった項目が存在している状態である．このような場合，大枠になる項目を親項目，中に含まれる項目を子項目というように定義する．1 つの親項目に対して子項目は複数存在する場合もあれば，1 つも存在しない，つまり親項目だけで 1 つの項目を表す場合もある．そのため，項目全体の名前としては親項目の名前を用いるものとする．つまり，基本的な項目というのは親項目のみで構成されていると言い換えることもできる．子項目が存在する場合は図 6 のように YAML 記述を行う．”child”キーのハッシュ値として項目の要素，つまり，ハッシュのリストを設定する．子項目中でパーツの指定等は行っているので，親項目の方ではパーツに関するハッシュは記述しない．また，この表形式フォームにおいて，項目内の 1 行に 2 つ以上のパーツを置きたい場合がある．図 5 中の郵便番号の項目であったり，図 8 のようにラジオボタンとテキストフィールドを並べて配置する項目等が例として挙げられる．この場合，親や子の項目の YAML 記述に 2 つ以上のパーツに

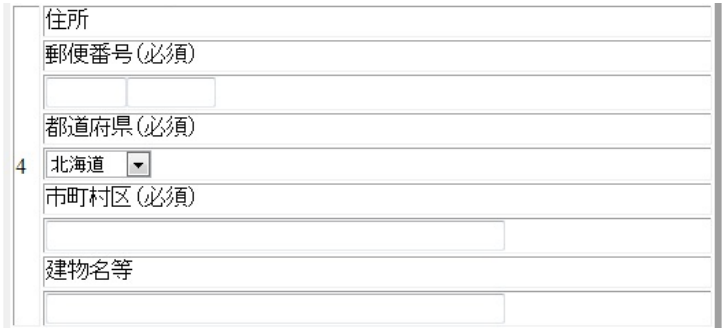


図 5　表形式フォーム例の一部（子項目の例）

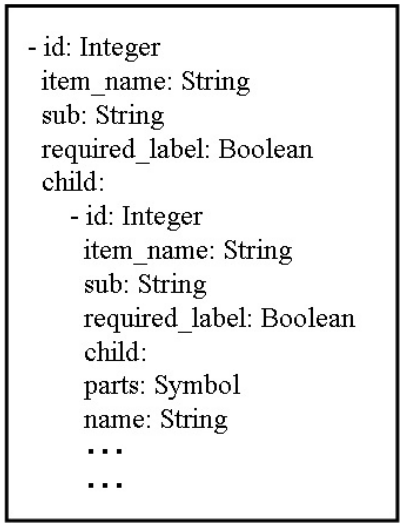


図 6　項目の YAML 記述（子項目あり）

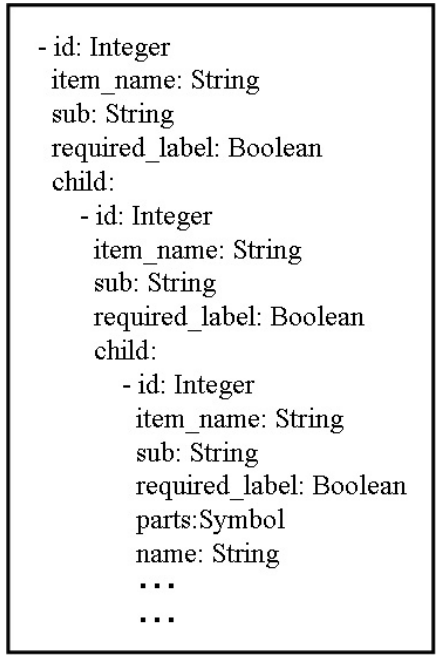


図 7　項目の YAML 記述（孫項目あり）


図 8　表形式フォーム例の一部（孫項目の例）

関するハッシュ内容を含めるようにはせず，それらのパーツをそれぞれ孫項目として設定する．孫項目は子項目の中に存在する形で定義する．孫項目を設定する場合の YAML 記述は図 7 のようになる．子項目内の”child”キーのハッシュ値として孫項目の要素となるハッシュのリストを記述する．子・孫の記述の場合も親・子のときと同様にして，子項目の方ではパーツに関するハッシュの記述は行わない．また，このフォームシステムにおいては項目の構造は親・子・孫の深さが 3 までとするので，孫項目のハッシュのリストには”child”キーに関する記述は含まないものとする．

## 3. インシデントレポートのフォーム生成とフォームデータの保存

まず，YAML 記述されたフォーム記述データからインシデントレポートのフォームの生成を行い，表示されたフォームの確認を行った．次に，生成したフォームに適当に入力を行って送信したフォームデータが params ハッシュテーブルに入っていることを確認したのち，それを取り出して，用意した Rails のモデルに保存した．

今回のフォーム生成とフォームデータの保存を行うシステムは，Ruby と Ruby on Rails を用いて作成した．使用した Ruby のバージョンは 1.9.2 で，Rails はバージョン 3.0.3 である．また，フォームを表示するブラウザには Internet Exproler8 を使用しており，フォームデータの確認には SQLiteman を使用した．

### 3.1 フォーム記述データから表形式フォーム生成

今回使用するインシデントレポートのフォーム記述データは，全て YAML 形式のファイルとして保存し，C:/sample というフォルダを作成してそこに入れておいた．九州大学病院の診療経過等報告システムにおけるインシデントレポートのフォーム数は 27 個であったので，YAML ファイルも incident1.yml～incident27.yml という名前で 27 個作成した．

システムの中心となる Rails アプリケーションとして incident_form というものを作成し，コントローラーは main というものを 1 つ用意した．生成するフォームは 27 個あるので，それぞれのフォームに対応させるためにビューファイルを 27 個準備した．

```
<html><head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
</head>

<%= create_page(p_id) %>

</html>
```

図 9　ビューファイル

このビューファイルはそれぞれ incident1.html.erb～incident27.html.erb という名前の eRuby ファイルとして作成されており，これらを使用できるようにするため，コントローラーのファイル内に同じ名前のメソッドをそれぞれ定義した．また，YAML ファイルを読み取り，そこからフォームを表示するための HTML 記述に変換するためのプログラムは Ruby で作成を行った．そのプログラムはそのまま main_helper.rb というヘルパーファイルに組込み，この Rails アプリケーションの中でヘルパーモジュールとして使用できる状態にした．

図 9 ではビューファイルの内容を示している．内容を簡潔に，また，このファイルを見ただけでは何が行われているかわかりにくくする意味も込めて，メソッドの記述 1 行でフォーム 1 つの生成を行うようにした．create_page はヘルパーモジュール内で定義されているメソッドであり，フォーム 1 つ分の HTML 記述をまとめる機能を持つ．p_id は引数であり，この部分にはそれぞれのフォームを指定する整数値が入る．つまり，ビューファイルは全てこの内容で統一されており，p_id の部分のみが異なるだけとなっている．そのため，p_id の値でフォームを選択できるように，フォーム記述データの YAML 記述の先頭部分に p_id に関するハッシュの記述を付加した．

生成される 27 個のフォーム全てが，項目の構造やその表示が崩れることなく，表形式フォームとして表現できるかどうかを，1 つずつブラウザに表示して確認した．確認手順としては，まずコマンドライン上で，incident_formアプリケーションのフォルダにてWEBrickサーバーを起動し，そしてブラウザで「http://localhost:3000/main/メソッド名」にアクセスした．例えば，incident1.html.erbを表示したい場合には「http://localhost:3000/main/incident1」へアクセスする．その結果，全てのフォーム記述データについて，このシステムを通して問題なくフォームを表示することができた．図 10～14 に生成したインシデントレポートのフォームの例をいくつか示す．

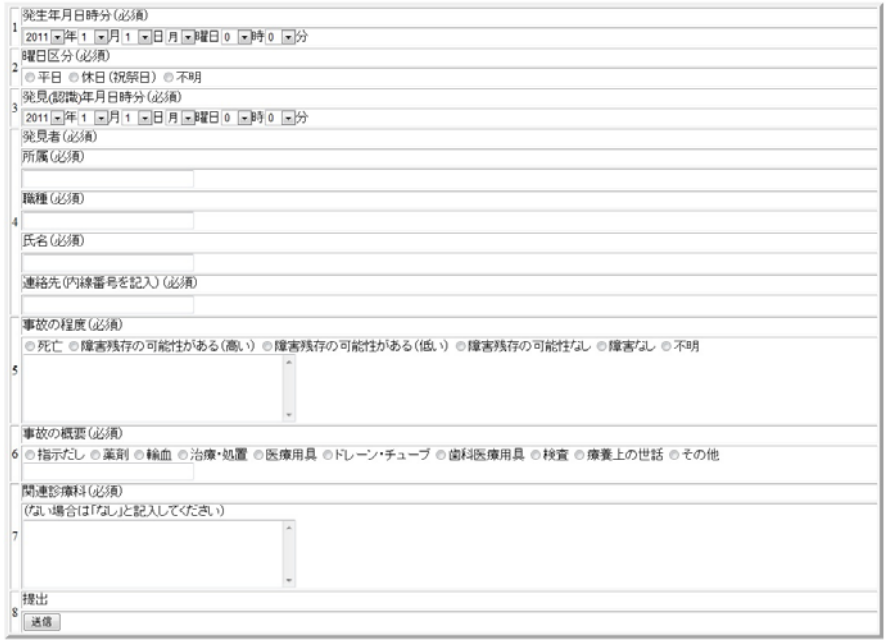


図 10　生成されたインシデントレポートのフォーム（incident1.html.erb）


図 11　生成されたインシデントレポートのフォーム（incident5.html.erb）

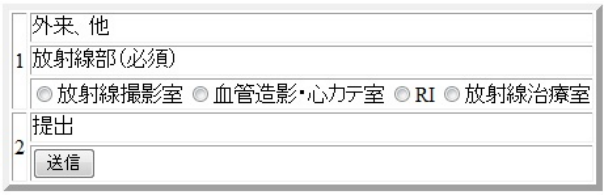


図 12　生成されたインシデントレポートのフォーム（incident7.html.erb）


図 13　生成されたインシデントレポートのフォーム（incident19.html.erb）


図 14　生成されたインシデントレポートのフォーム（incident27.html.erb）

### 3.2 送信されたフォームデータの保存

ここでは，フォームデータ保存用に少し変更を加えながら，システムとして引き続き 3.1 で作成した incident_form アプリケーションを利用した．

まず，送信ボタンを押した際に何らかのアクションが起きるようにするため，form_tag メソッドを用いてビューファイルを少し書き直した．図 15 にその内容を示す．赤枠で囲った部分が追記したものとなる．指定したアクション名に関しては後述する．次に，フォームデータを保存するために FormData という Rails のモデルを生成し，デ

ータベースのテーブルとして form_datas を作成した．このテーブルには，自動で設定される属性の id と created_at，updated_at 以外に 2 つの属性を用意した．それは，どのフォームから送信されたフォームデータかを識別するために p_id の値を入れる pid と，フォームデータを入れる fd_hash である．Rails では，送信されたデータは params ハッシュテーブルの中に一時的に格納されるので，そのハッシュテーブルをフォームデータとして保存することにした．ただし，ハッシュテーブルはそのままの状態で保存するのではなく，JSON 文字列に変換してテキストデータとして保存した．そのため，fd_hash は TEXT 型に設定した．また，params をそのまま保存してしまうと，フォームデータとは関係の無い情報が含まれてしまうので，その部分は取り除いてフォームデータだけを取り出すことにした．main コントローラーには，フォームデータの保存に関するメソッドとして fd_sample を追記し，これをフォームデータが送信されたときのアクションとして，ビューファイルの form_tag メソッド内にて指定した．また，fd_sample.html.erb というビューファイルも作成し，params の中身と JSON 文字列に変換したフォームデータがブラウザ上にも表示されるようにし，変換した JSON 文字列を逆変換したものも表示することで，正しく変換されていることも目視できるようにした．

フォームデータの保存は，まず，3.1 におけるフォームの表示の確認手順と同様にして WEBrick を起動し，インシデントレポートのフォームを表示してから適当に入力を行って送信ボタンを押す．そして表示された送信内容や JSON 文字列と，SQLiteman で表示したテーブルの中身を比較して，正しく保存が行われているかどうかを見た．例として図 16 に，最初に「http://localhost:3000/incident4」へアクセスして入力を行って，送信ボタンを押したときに表示された内容を示す．図 17 は，そのときの form_datas テーブルの中身の一部を示している．また，全てのフォームにおいて一連の操作を行ったあとの form_datas テーブルの中身の一部を図 18 に示す．27 個のフォームから送信したフォームデータを期待通りに保存することが出来た．

## 4. おわりに

九州大学病院の診療経過等報告システムのインシデントレポートを参考に，フォームを構成する要素であるフォーム記述データを YAML 形式で構造記述した．そして，そのデータから表形式で表現されるフォームを生成するようなシステムを，Ruby と Ruby on Rails を用いて作成した．27 個のインシデントレポートのフォームに関するフォーム記述データを用意し，このシステムを通して生成されたフォーム全てが，項目の構造やその表示が崩れることなく，表形式フォームとして表現できることをブラウザ上で確認したところ，全てのフォームが問題なく表示された．また，インシデント

```
<html><head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
</head>
<% form_tag ({:action => 'アクション名'}) do %>
<%= create_page(p_id) %>
<% end %>
</html>
```

図 15　ビューファイル

**Form Data Display**

○フォームデータ

{"syozoku_sinnryouka"=>"第一外科", "houkoku_name"=>"近藤浩志", "houkoku_num"=>"1234", "submitting"=>"送信"}

○JSON文字列に変換

{"syozoku_sinnryouka":"\u7b2c\u4e00\u5916\u79d1","houkoku_name":"\u8fd1\u85e4\u6d69\u5fd7","houkoku_num":"1234","submitting":"\u9001\u4fe1"}

○JSON文字列を逆変換

{"syozoku_sinnryouka"=>"第一外科", "houkoku_name"=>"近藤浩志", "houkoku_num"=>"1234", "submitting"=>"送信"}

図 16　送信したフォームデータの表示

| | id | pid | fd_hash |
|---|---|---|---|
| 1 | 1 | 4 | {"syozoku_sinnryouka":"¥u7b2c¥u4e00¥u5916¥u79d1","houkoku_name":"¥u8fd1¥u85e4 |

図 17　form_datas テーブル

| | id | pid | |
|---|---|---|---|
| 2 | 2 | 1 | {"occur_year":"2011","occur_month":"6","occur_day":"15","occur_youbi":"¥u6c34","occur_hour":"12","occur_minut |
| 3 | 3 | 3 | {"toujisya1_name":"¥u4e5d¥u5927¥u592a¥u90ce","toujisya_syokusyu":"¥u770b¥u8b77¥u52a9¥u624b","toujisya_ |
| 4 | 4 | 5 | {"occur_gairai":"¥u533b¥u7642¥u5916¥u6765¥u203b","occur_byoutou":"¥u624b¥u8853¥u90e8","occur_other_cor |
| 5 | 5 | 2 | {"patient_name":"¥u4e5d¥u5927¥u82b1¥u5b50","patient_id":"abcd1234","patient_age":"24","patient_age_month |
| 6 | 6 | 6 | {"iryou_gairai":"¥u5916¥u6765¥u5f85¥u5408¥u5ba4","iryou_gairai_other":"","submitting":"¥u9001¥u4fe1"} |
| 7 | 7 | 7 | {"housyasenbu":"¥u653e¥u5c04¥u7dda¥u6cbb¥u7642¥u5ba4","submitting":"¥u9001¥u4fe1"} |
| 8 | 8 | 8 | {"sika_gairai":"¥u5916¥u6765¥u8a3a¥u7642¥u5ba4","sika_gairai_other":"","submitting":"¥u9001¥u4fe1"} |
| 9 | 9 | 9 | {"sika_housyasenbu":"¥u653e¥u5c04¥u7dda¥u64ae¥u5f71¥u5ba4","submitting":"¥u9001¥u4fe1"} |
| 10 | 10 | 10 | {"byoutou_center1":"¥u30ca¥u30fc¥u30b9¥u30b9¥u30c6¥u30fc¥u30b7¥u30e7¥u30f3","byoutou_center1_other":" |
| 11 | 11 | 11 | {"byoutou_north6":"¥u51e6¥u7f6e¥u5ba4","byoutou_north6_other":"","submitting":"¥u9001¥u4fe1"} |
| 12 | 12 | 14 | {"byoutou_north10":"¥u75c5¥u5ba4","byoutou_north10_other":"","submitting":"¥u9001¥u4fe1"} |
| 13 | 13 | 13 | {"byoutou_north9":"¥u305d¥u306e¥u4ed6","byoutou_north9_other":"¥u30c6¥u30b9¥u30c8¥u3067¥u3059","subm |
| 14 | 14 | 12 | {"byoutou_north7":"¥u6d74¥u5ba4","byoutou_north7_other":"","submitting":"¥u9001¥u4fe1"} |
| 15 | 15 | 15 | {"byoutou_north11":"¥u30c8¥u30a4¥u30ec","byoutou_north11_other":"","submitting":"¥u9001¥u4fe1"} |
| 16 | 16 | 16 | {"byoutou_houso":"¥u5eca¥u4e0b","byoutou_houso_other":"","submitting":"¥u9001¥u4fe1"} |
| 17 | 17 | 17 | {"byoutou_east1":"¥u305d¥u306e¥u4ed6","byoutou_east1_other":"¥u30c6¥u30b9¥u30c8¥u3067¥u3059¥uff12","s |
| 18 | 18 | 18 | {"byoutou_east2":"¥u968e¥u6bb5","byoutou_east2_other":"","submitting":"¥u9001¥u4fe1"} |
| 19 | 19 | 19 | {"byoutou_south4":"¥u900f¥u6790¥u5ba4","byoutou_south4_other":"","submitting":"¥u9001¥u4fe1"} |
| 20 | 20 | 27 | {"jikotyokuzen":["¥u8074¥u899a¥u969c¥u5bb3","¥u8a8d¥u77e5¥u75c7¥u30fb¥u5065¥u5fd8","¥u6b69¥u884c¥u96 |

図 18 `form_datas` テーブル

レポートのフォームを表示してから適当に入力を行ってフォームデータを送信し，表示された送信内容やJSON文字列とSQLitemanで表示したテーブルの中身を比較して，正しく保存が行われているかどうかを見た．その結果，27個のフォームから送信したフォームデータを期待通りに保存することが出来た．

**謝辞** 九州大学病院診療経過等報告システムの共同研究開発者であり，本研究に必要となる各種のフォームについての実例として，九州大学病院診療経過等報告システムのフォームについての具体的な情報をご提供下さった九州大学医療安全管理部・セーフティマネージャの秋好美代子看護師に心より感謝致します．