スプレッドシートの構造からDB設計を行うシステム開発

System development by following spreadsheet

西野嘉之†‡
Yoshiyuki Nishino
†産業能率大学
‡ユーレット株式会社
†SANNO UNIVERSITY.
‡Ullet Co., Ltd.

要旨

業務を行う際、Excel ファイルなどのスプレッドシートで帳票フォーマットを作成し、業務を行っている。しかしながら、データベース（DB）に蓄積するシステムを構築しようとすると、新たに画面設計や DB 設計を行わなければならない。そこで、本研究ではスプレッドシートのレイアウトをデータベース構造とみなし、自動的に DB 設計を行い、DB を生成しデータを蓄積する方法論を提案する。さらに、Excel の入力ファイルから DB の自動生成を実証し、その効果を示す。その結果、値に対して入力フォーマットの位置情報と DB 処理情報と出力フォーマットの位置情報を紐付けることで、Robotic Process Automation（RPA）によって、共生することができる情報システムを構築できることを明らかにする。

1. はじめに

PC やタブレット端末の普及により、ビジネスドキュメントを Excel で作成する機会が多くなり、情報資源がファイルによって蓄積されている。また、データを分析するデータサイエンティストという職種が注目を集めている。一方、会計システムや販売管理システムなどの基幹システムを構築し、データベース (DB)に蓄積されたデータを情報資源として管理し、企業の情報戦略のための Information として活用している[1]。このように、ファイル単位で管理されているデータと、システムが構築され、DB に蓄積されているデータというように、大きく分けて 2 つの保存形式が存在する。

しかしながら、ファイル単位で保存されているデータを集計したり、一覧にしたり、特定条件で抽出したりすることを求められる際には、DB に格納しなければならない。そのために、システム構築を行うが、ユーザインタフェースとなる Excel などのスプレッドシートのレイアウトや、入力するデータ項目などが頻繁に変わる場合、システムの仕様変更を行うため、DB 設計から見直すことになり運用コストの増大や、メンテナンスの柔軟性が低いことが問題となる。従って、Excel などのスプレッドシートを情報資源としてそのまま活用するには、ファイルの関係性や位置情報などを保持したまま DB に格納し、集計、一覧化及び条件による抽出が可能となるシステムが求められる。これまでに概念帳票といった研究や、CSV ファイルなどでインポートする方法などは提案されているが、スプレッドシートの構造をそのまま情報資源として DB に格納する方法は実現されていない[2]。また、画面設計を行えば DB が自動生成される手法もあるが、この手法は既に使用している Excel などのスプレッドシートを活用することができず、一から画面を構築しなければならない[3]。

そこで本研究では、「Book」、「Sheet」「Cell」の位置情報を DB のデータ構造とみなし、DB に格納することで、DB 設計を行うことなく、DB を構築することができる手法を提案する。さらに、エンジニアの人材評価を行うためのスキルシートである Excel ファイルを用いて、DB 構築を行うと同時に、データを格納することができることを実装する。その結果、Excel などのスプレッドシートを元に、DB 設計を行うことなく、DB 構築を行い、尚且つファイル構造を元にすることで、自動的にデータが格納される手法の有効性を示す。さらに、入力データからデータ分析をするために再集計や集計する変数を容易に変更することができることにより、ユーザはデータの検証をしながらシステムを成長させることができることを示す。最後に、情報システムと共生するには、ユーザの思考に従ってシステムが変更することができ、ユーザが入力したデータがどのように DB に格納され、どのように出力されているかを、ユーザ自身の思考と一致するように、情報を紐づけることが重要であることを明らかにする。

2. 従来モデル

本研究では、要件定義を行い、基本設計及び DB 設計を行う、ウォーターフォール型の開発手法を従来モデルとする。従来の開発手法では、ユーザが普段使っている帳票を元に入力画面と出力画面を作る。この方法の問題点は、プログラミングや DB 構築を開始するフェーズに入る前に、入力画面と出力画面を確定させ、DB 設計を行う。従って、構築後に仕様変更を行いたい場合には、DB 構造に影響がでるため、容易に変更することができない。このような問題は、ユーザが Excel などのスプレッドシートで業務を運用している場合、システム構築を行った後、自分たちがイメージしていた画面や帳票ではないという齟齬が生じる。なぜなら、Excel などのスプレッドシートには、「「Book」、「Sheet」「Cell」という、データの関係性と位置情報というデータ構造がすでにユーザの手によって構築されているにもかかわらず、システム構築を行おうとすると、DB 設計者は、ユーザが使用する入力画面や帳票のデータの関係性や位置情報を考慮せず、DB という概念の中で、新たにデータ構造を構築するからである。図 1 に、従来の帳票設計と DB 設計の関係を示す。

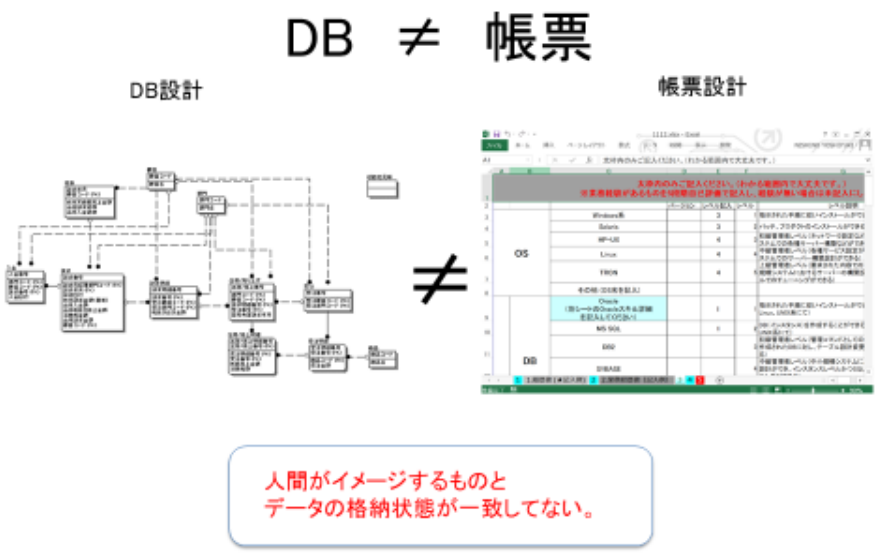

図 1 ：従来の帳票設計と DB 設計の関係

3. 提案方法

「BOOK」、「Sheet」「Cell」の位置情報を DB のデータ構造とみなし、DB に格納することで、DB 設計を行うことなく、DB を構築できる手法を提案する。

3.1. ユーザ思考＝DB

図 2 に示すように、ユーザが作成した Excel などのスプレッドシートの構造を元に、DB を生成してデータを格納する。これにより、データベースの設計をすることなく、DB 構造を生成することが可能となる。同じフォーマットの Excel などのスプレッドシートがあれば、スプレッドシートを入力画面と見なし、入力されたデータを DB に格納することができる（式 1）。ユーザが見ている情報は DB ではなく、入力ファイルである。従って、ユーザが入力ファイルと DB 構造が一致していれば、ユーザは DB 内にどんな値がどこにどのような状態で格納されているかを容易に認識することができる。

値の場所＝　C: ¥[ファイルの保存先] ¥[ファイル名（Book）] ¥[Sheet 名] ¥[セル名]　　(1)

さらに、入力ファイルの追加、変更、削除という行為と、DB 構造と値の変更を直接連動させることで、ユーザの行為に従って、DB を成長させることが可能となる。図 3 に示すように、既存のスプレッドシートを使用したま

まシステム構築を行い、運用して変更点があれば入力ファイルや出力条件をユーザが変更することで、システムを修正することが可能となる運用サイクルを実現することができる。

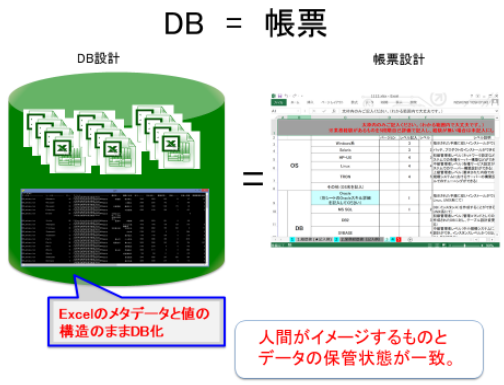

図２：入力レイアウトと DB 構造が一致する関係

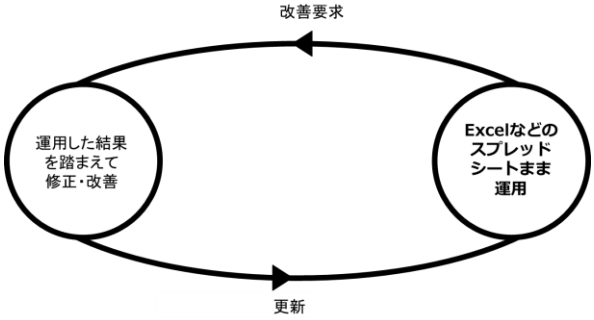

図３：持続可能なシステム構築のサイクル

3.2. スプレッドシートのデータ構造から DB 設計

図 4 に示すように、DB 内にデータを格納するための構造は、「Book」の中の「Sheet」の「Cell」と特定できるようにデータを格納する。この時、データの位置情報を絶対パスとして値に紐付けて格納しておく。さらにデータ出力の効率化を図るために、行あるいは、列でグループを作ることで、データをまとまった状態で格納し、出力ファイルを生成する際には、データのグループ単位で抽出する。図 5 に示す。

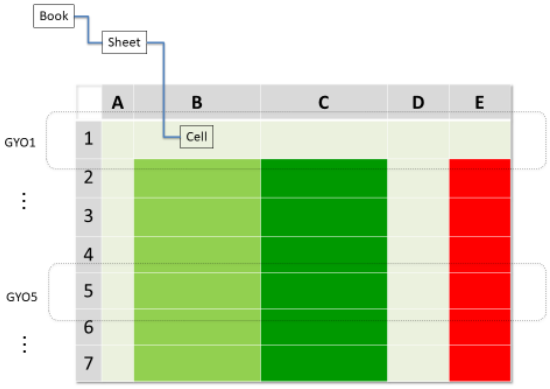

図 4：DB 生成とデータ格納の概念イメージ

図 5：DB からデータを条件抽出する概念イメージ

3.3. 複数の同一テンプレートファイルを DB 化

図 6 に提案モデルの運用プロセスを示す。

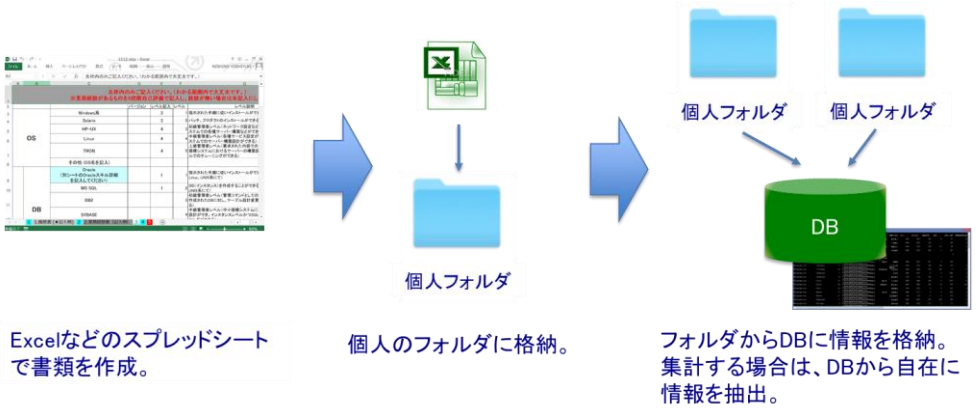

図６：　提案モデルの運用プロセス

Excel ファイルなどのスプレッドシートの帳票テンプレートに情報を入力し、フォルダ内に保存する。フォルダ

に保存されたファイルから自動的に DB 生成を行い、データを格納する。

4. 構築結果

ユーザが作成したスキルシートの構造を元に、DB を生成してデータを格納し、出力ファイルを生成する。

4.1. 入力ファイル

エンジニアの人材評価を行うためのスキルシートを用意する。エンジニア「1112」と「1113」のスキルシートをサンプルファイルとして、入力ファイルとする。図 7 は 1112.xlsx のスキル詳細 Sheet であり、図 8 は 1112.xlsx の ORACLE スキル詳細 Sheet を示す。

図 7：1112.xlsx のスキル詳細「Sheet 3」

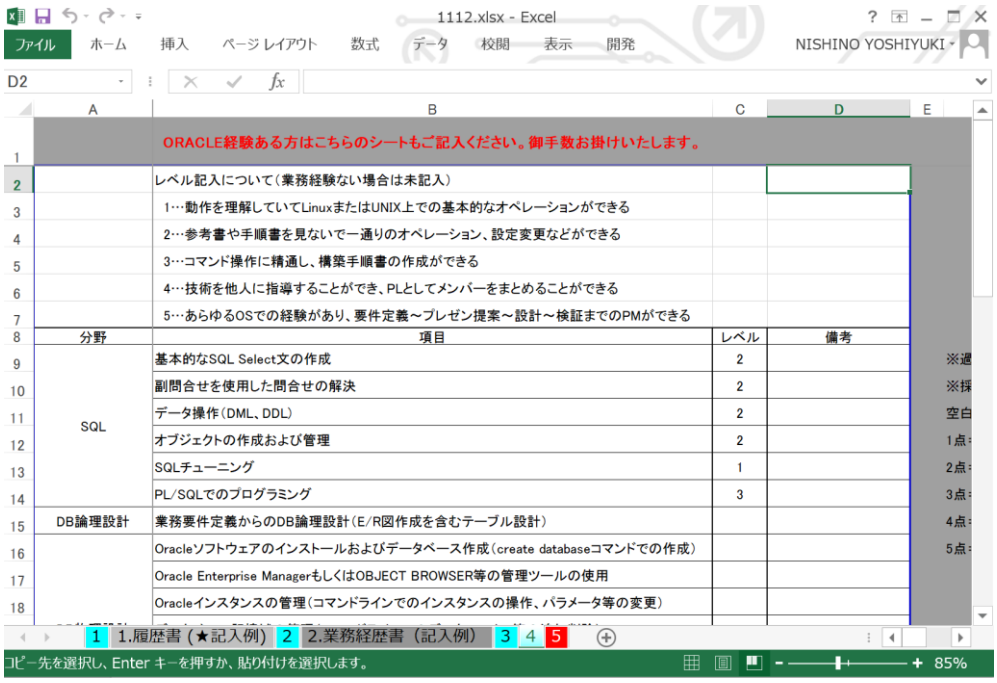

図 8：1112.xlsx の ORACLE スキル詳細「Sheet 4」

4.2. 自動的に DB を生成しデータを格納

図 6 に示したプロセスのように、Excel ファイルを、フォルダに格納し、DB にデータをアップロードする。Excel ファイルをアップロードすると、ファイルのデータ構造を元に DB を自動生成すると同時に、値も自動的に格納する。データを格納する際のデータ構造を図 9 に示す。図 9 のようにデータを取り出す工程は、図 4 のように、Excel ファイルの「Book」「Sheet」「Cell」の関係性をデータ構造として、格納する。

Book	Sheet	GYO	Cell	Value
人事.xlsx	スキル詳細	1	A1	太枠内のみご記入ください。（わかる範囲内で大丈夫です。）※業務経験があるものを5段階自己評価で記入し、経験が無い場合は未記入にしてください。
		2	A2	Null
			B2	Null
			C2	Null
			D2	バージョン
			E2	レベル記入
			G2	レベル

図 9:　データを格納した際のデータ構造

4.3. 出力ファイルを自動生成

次に、出力ファイルを Excel Sheet として生成した結果を図 10 に示す。図 7 : 1112.xlsx のスキル詳細「Sheet 3」の「Book」名と Cell の「C 列」と「レベル」列を、「氏名 ID」「スキル詳細」「スキルレベル」として、図 10 に値を抽出した。さらに、図 8 : 1112.xlsx の ORACLE スキル詳細「Sheet 4」の「項目」列と「レベル」列から、「ORACLE スキル詳細」と「ORACLE スキルレベル」として、図 10 に値を抽出した。さらに、1113 のエンジニアのスキル詳細「Sheet 3」と ORACLE スキル詳細「Sheet 4」もマージし、1 つの Excel Sheet に出力した CSV ファイルは、列名＝データ項目名となり、データの関係性は列でしか保持されないが、提案手法の出力ファイルは、セルの位置情報によってデータの関係性が保持されているため、出力ファイルを自動生成する場合でも、データの位置情報を用いている。

図 11 にデータの値とその場所を示す情報を付加して、「SQL」と「Java」という Word が、「スキル詳細」の列に存在するデータを検索し、抽出した例を示す。例えば、「1112」の「スキル詳細」の「SQL」は、人材評価シート 1112.xlsx の「¥¥C:¥QMUSERS¥NISHINOSAN2¥SkillSheet¥1112.xlsx#'3'!C35」として、格納されている。「C35」は、Excel シートのセルの位置を表している。つまり、図 11 の「SQL」は、人材評価の Excel ファイルの、「Book」＝1112 の「Sheet」＝３の「Cell」＝C35 に入力されている値で、DB には、値＝SQL とその在りかを示している絶対パス「¥¥C:¥QMUSERS¥NISHINOSAN2¥SkillSheet¥1112.xlsx#'3'!C35」という情報が紐付いて格納されている。提案手法では、入力ファイルのどのセルに値が入っていて、どのようなレイアウトでファイルが構成されているかを元に DB のデータ構造を構築することで、入力ファイルが持つデータの関係性を引き継ぎ、さらに出力ファイルを生成した際には、入力ファイルのどの値を用いて計算されたのかを保有している。従って、「1112」の「スキルレベル」＝11 は、「SQL」と「Java」の合計点数を表しており、1112.xlsx のシート「３」の「C10」＝1、C34＝3、C35＝3、C51＝4 は、合計 11 点の内訳を示している。また、図 10 の「1112」の列名「スキル詳細」の「SQL」と「Java」の Word が含まれる列名「スキルレベル」の点数を見てみると、図 11 の「SQL」と「Java」の内訳と合致する。

Sheet 3

氏名ID	スキル詳細	スキルレベル
1112	Windows系	3
	Solaris	3
	HP-UX	4
	Linux	4
	TRON	4
	Oracle	1
	MS SQL	1
	Java	3
	SQL	3
	Perl	3
	Java	4
**		33
1113	Windows系	3
	Solaris	3
	HP-UX	4
	Linux	4
	TRON	4
	Oracle	1
	MS SQL	1
	Java	3
	SQL	3
	Perl	3
	Java	4
**		33
		66

Sheet 4

ORACLEスキル詳細	ORACLEスキルレベル
基本的なSQL Select文の作成	2
副問合せを使用した問合せの解決	2
データ操作（DML、DDL）	2
オブジェクトの作成および管理	2
SQLチューニング	1
PL/SQLでのプログラミング	3
STATSPACKを使用した、レポートの作成と、解析作業	1
SQL SERVER、MYSQL、SYBASE、DB2等の他DBからORACLEへのデータ移行作業	2
	15
基本的なSQL Select文の作成	1
副問合せを使用した問合せの解決	1
データ操作（DML、DDL）	1
オブジェクトの作成および管理	1
SQLチューニング	1
PL/SQLでのプログラミング	2
データベース・リカバリ	1
フラッシュバック・データベース（10g）	1
ユーザー・エラーからのデータリカバリ（archivelogを使用）	1
データベース破損の処理（バックアップファイルからのDB補修作業）	1
STATSPACKを使用した、レポートの作成と、解析作業	1
データベースの監視（V$表を使用した適切な監視とパラメータの修正）	2
アドバイザの使用（OEM）	2
	16
	31

図 10：スキル詳細及び ORACLE スキル詳細の出力ファイル

Sheet 3

氏名ID	スキル詳細	スキルレベル	スキルSHEET
1112	MS SQL	11	¥¥C:¥QMUSERS¥NISHINOSAN2¥SkillSheet¥1112.xlsx#'3'!C10
	Java		¥¥C:¥QMUSERS¥NISHINOSAN2¥SkillSheet¥1112.xlsx#'3'!C34
	SQL		¥¥C:¥QMUSERS¥NISHINOSAN2¥SkillSheet¥1112.xlsx#'3'!C35
	Java		¥¥C:¥QMUSERS¥NISHINOSAN2¥SkillSheet¥1112.xlsx#'3'!C51
1113	MS SQL	11	¥¥C:¥QMUSERS¥NISHINOSAN2¥SkillSheet¥1113.xlsx#'3'!C10
	Java		¥¥C:¥QMUSERS¥NISHINOSAN2¥SkillSheet¥1113.xlsx#'3'!C34
	SQL		¥¥C:¥QMUSERS¥NISHINOSAN2¥SkillSheet¥1113.xlsx#'3'!C35
	Java		¥¥C:¥QMUSERS¥NISHINOSAN2¥SkillSheet¥1113.xlsx#'3'!C51
		22	

図 11：「SQL」｛Java｝のキーワードによる出力

5. おわりに

本研究では、ユーザが Excel ファイルなどのスプレッドシートなどで、既に作成している帳票フォーマットなどをそのまま情報資源として捉え、新規に業務システムのための入力画面を作成することなく、さらにスプレッドシートの「Book」、「Sheet」及び「Cell」の位置情報をデータ構造として定義することで、DB 設計を行うことなく DB を構築することができる方法を提案した。具体的な例として、人材評価の帳票テンプレートのレイアウトを元に DB 生成を自動的に行い、さらにテンプレートに入力されている値を DB に自動的に格納できることを実装し、確認した。さらに、構築した DB から条件指定したデータを抽出し、自動的に出力ファイルを生成できることを確認した。また、ユーザが作成したファイル名のスプレッドシートのデータと、DB のデータ構造と、そこから条件抽出した出力ファイルのデータの関係性が紐付いていることを確認した。その結果、値に対して入力フォーマットの位置情報と DB 処理情報と出力フォーマットの位置情報を紐付けることで、RPA によって、共生することができる情報システムを構築できることを明らかにした。

参考文献

[1] 松平和也, “IRM-情報資源管理のエンジニアリング”, 日経 BP 社, 1990, Aug.
[2] 中西昌武, “データソース・タイプ，概念帳票生成パターン，および概念帳票テンプレートのカタログ化”, 経営情報学会誌, Vol.11, No.1, 2002. June.
[3] 西野嘉之, “DB 設計を行わないシステム開発”, 情報システム学会誌, 第 13 回全国大会・研究発表大会, 2017.