一個以關連式資料庫爲基礎的企業規則庫開發策略

程裕繁、裘錦天、姜國輝

摘要

將隱匿在程式碼之中的零散企業規則抽取出來，予以整合成一個獨立的企業規則庫，如此可以大幅度增加資訊系統的企業規則獨立性，進而有效改進資訊系統的可維護性。本文提出一個以關連式資料庫程序爲基礎的企業規則庫開發策略，首先提出一個具有簡單、便宜、高跨平台等優點的企業規則庫架構，然後再進一步提出一個企業規則庫的開發程序。

關鍵字：企業規則、企業規則獨立性、企業規則庫、關連式資料庫、資料庫程序

程裕繁：修平技術學院資訊管理系講師、政治大學資訊管理系博士生
裘錦天：政治大學資訊管理系副教授
姜國輝：政治大學資訊管理系副教授

A RDB-Based Business Rule Base Development Strategy

Bill Y.F. Cheng, Jiin-Tian Chyou, Johannes K.H. Chiang

Abstract

A business rule base (BRB) is a collection of related business rules that can improve the business rule independence which can be defined as the capacity to change the business rules without having to change the information system. This paper presents a RDB-based strategy (architecture and development process) to support the development of BRB.

Keywords: Business Rule, Business Rule Independence, Business Rule Base, Relational Database, Database Procedure

Bill Y.F. Cheng:Lecturer, Department of MIS, Hsiuping Institute of Technology
Doctoral Student, Department of MIS, Chengchi University
Jiin-Tian Chyou: Associate Professor, Department of MIS, Chengchi University
Johannes K.H. Chiang: Associate Professor, Department of MIS, Chengchi University

一、前言

資料庫技術的進步讓企業的資訊系統具備了資料獨立性（Data Independence）。在大部份的情況下，資料庫結構的更動並不會影響到資訊系統，資訊系統並不需要配合著去修改任何的程式碼[2]。

相較於資料管理方面的受到重視與進步，同樣是資訊系統重要元素之一的企業規則（Business Rules），其受到重視的程度與管理技術的進展則相對顯得不足與不成熟。對於大多數的資訊系統來說，企業規則依然是隨意地被散佈在資訊系統的許多地方，也仍然是隱晦地被使用在程式碼之中。

其實，如果能夠像資料一般地將企業規則從程式碼之中抽離出來，予以整合成一個獨立的企業規則庫（Business Rule Base），如此可以讓資訊系統具備良好的企業規則獨立性（Business Rules Independence），降低企業規則變動對資訊系統的影響，進而改進資訊系統的可維護性[6,7,8]。

本文提出一個以關連式資料庫程序爲基礎的企業規則庫開發策略，內容包含一個企業規則庫架構，以及一個企業規則庫的開發程序。

全文共分成六節。除了本節之外，第二節是企業規則與企業規則庫的定義，第三節探討各種提升企業規則獨立性的解決方案，第四節談企業規則庫的架構，第五節解釋企業規則庫的開發程序，第六節則是本文的結論。

二、企業規則與企業規則庫

企業規則所指的是，對企業的某一方面做定義或限制的文字性陳述，其目的在於清楚表明企業的結構，或是用以控制或影響企業的行爲[1]。若予以詳細區分，企業規則可以分成限制（Constraints）、指引（Guidelines）、行動促成（Action Enablers）、計算（Computations）等四種[8]。

限制是一種必需予以遵守的規則，例如客戶的訂購金額不可以超過該客戶的信用餘額。指引是一種警告性的規則，例如資訊管理師應該具有大學以上學歷。行動促成是一種在某一個條件成立時必需予以遵守的規則，例如如果客戶的單次訂購金額超過三千五百元則予以核發貴賓卡一張。計算是一種用來說明特定用途公式的規則，例如薪資的計算公式。

在純人工營運的企業中，企業規則散佈在各種的企業文件之中，或是存放在某些人的腦袋之中。而在資訊化營運的企業中，企業規則也實作在資訊系統當中。這些企業賴以生存的企業規則應該用文字予以詳細記錄下來，並定立一套正式的制度來予以妥善管理。

另一方面，如果企業規則被隨意地以隱晦的方式在程式碼之中實作，那麼將造成維護上的困難。首先，我們不知道那些程式碼是企業規則的實作，要修改就要用人工去慢慢找出來，不祇是很浪費人力，也很容易的就會漏改。還有，同一個企業

規則通常會有多個實作，而這些實作很有可能因爲人或時間的因素而造成了不一致的現象。再者，因應經營大環境的快速變遷，企業規則會不斷的翻新，將導致上述之維修困難的問題更加嚴重。

上述問題需要從兩個方向來解決。第一個方向是管理手段，亦即要做好企業規則的管理，記錄資訊系統所使用的每一條企業規則，也記錄企業規則的每一條實作。第二個方向是技術手段，亦即將資訊系統所使用的企業規則從資訊系統之中抽離出來，予以獨立儲存成一個企業規則庫，藉以大幅度提升資訊系統不受企業規則變動影響的能力（企業規則獨立性）。

良好的企業規則庫應該具備下列三項特性：

（一）完整性（Integrity）：企業規則庫所儲存的企業規則必定與事實相符（正確性），而且同一個企業規則的多份儲存都會完全相同（一致性）。

（二）共享性（Sharable）：企業規則庫必需能夠讓同一個資訊系統的不同部份（使用者介面、處理邏輯、資料庫），或是讓多個資訊系統在同一個時間一起使用。

（三）獨立性（Independence）：企業規則庫獨立於程式之外，因此在大多數的情況下，企業規則的變動將不會對程式造成影響。

三、企業規則獨立性的改善方案

爲了降低企業規則變動對程式的影響，開發人員一直致力於將企業規則獨立於程式之外。經由文獻的探討和對業界的觀察和訪談，大致上可以將目前的企業規則獨立性改善方案歸類成下列三種：

（一）將企業規則的規則參數（Rule Parameters）[1] 或是計算公式從程式中抽離出來，當做資料予以儲存在資料庫或檔案之中，而程式則視需要予以動態取出來使用（可以直接使用或是需要經過剖析、轉換等前置處理）。這種解決方案多數都有輔以良好的維護介面，讓使用者自己就能夠進行這些規則參數和計算公式的維護。

這種作法的效果不錯，也很容易實作，所以使用的很普遍。但是這種作法並沒有實際將企業規則從程式之中抽離出來，因此仍然無法不受較大幅度企業規則變動（不祇是修改規則參數的變動）的影響，也無法解決同一個企業規則有多個實作所造成的維護問題。

（二）將企業規則包裝成函數或是類別（可能是程式的一部份，也可能是獨立的函式庫或是類別庫），然後在程式

1 Morgan 稱規則參數爲企業參數（Business Parameters），並對其做了如下定義：企業參數是企業規則用來定義邊界，在列舉中辨識名稱，或是提供特定字串值的一種關鍵值（Critical Values）[6]。例如規則：客戶的最高信用額度是5000，其5000就是一個規則參數。又例如規則：本行承作放款客戶的年齡屆於18歲至65歲之間，其18和65都是規則參數。再例如規則：從事軍、公、教等三類職業的國民是本行積極承作的客戶群，其軍、公、教都是規則參數。

中叫用（invocation）這些企業規則的實作函數或是實作類別。

如果全面採用函數來實作所有的企業規則，並將這些函數集中成獨立的動態連結函數庫（Dynamic Linking Library），然後讓所有的程式都透過這個函數庫來執行企業規則。如此確實能夠讓程式完全獨立於企業規則之外，幾乎不受企業規則變動的影響。

可是這種全面採用的情況在實務上並不常見。而且這種作法和特定程式語言或是特定作業平台緊密相關，針對特定程式語言或是特定作業平台開發的規則函數庫並不容易共用於其它程式語言和作業平台，亦即這種作法的跨語言/平台的能力並不理想。

（三）以購買手段，使用商用的企業規則管理軟體來提升資訊系統的企業規則獨立性。

von Halle[8]將商用的企業規則管理軟體劃分成資料改變導向（Data-Change- Oriented）和服務導向（Service-Oriented）兩種。

資料改變導向軟體的主要用途在於減輕企業規則在資料庫上施行的困難度與作業負擔。這種軟體讓使用者使用特定的企業規則定義語言（Business Definition Language）來定義企業規則，然後自動將這些企業規則轉換成各種的資料庫完整性確保機制（Not Null, Check, Foreign Key, Trigger等），藉以讓資料庫遵守企業規則，確保資料庫的完整性。USoft, Versata等均屬此種軟體。

服務導向軟體的主要用途則在於減輕企業規則在程式（主要是針對處理邏輯部份）上實施的困難度與作業負擔。這種軟體讓使用者使用特定的規則語言來定義企業規則，然後使用特殊的程式設計技術（攔截並分析程式碼，或是做成程式模組與程式一起編譯和執行）來確保程式的不會違反企業規則。HNC, ILOG等均屬此種軟體。

目前商用的企業規則管理軟體往往祇針對部份的問題做解決，而且都與特定程式語言或是作業平台緊密相關，其實用性並不理想，仍然處於不成熟的初始階段[6]。

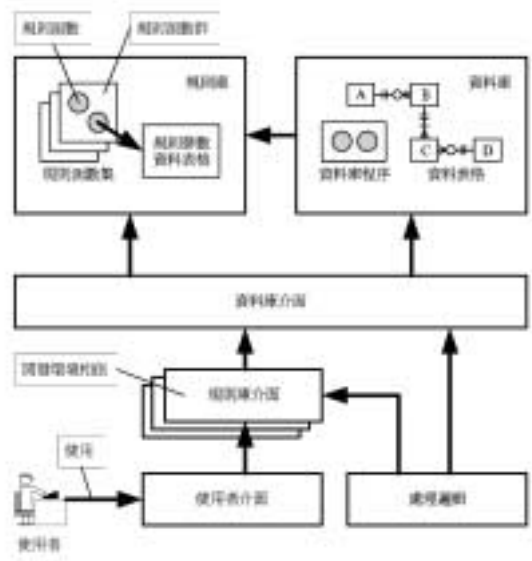

圖 1 企業規則庫架構

四、企業規則庫架構

整合上一節所說明的第一種作法（規則參數）與第二種作法（規則函數），再以關連式資料庫的優越特性來予以調整，本文提出一個簡單卻實用的企業規則庫架構（請參考圖1）。

這個架構仍然使用函數來實作企業規則，但使用的不再是特定程式語言（Java, Delphi, Visual Basic等）的函數或類別，而是改爲採用關連式資料庫的資料庫程序（Database Procedures, 如User Defined Functions, Stored Procedures, Triggers等均是）。

這些規則的實作函數將與資料庫放在一起，所以資料庫可以直接使用這些規則函數來遵守企業規則，確保資料庫的完整性。而在程式方面，不論是使用者介面還是處理邏輯部份，祇要具備有與資料庫連結的能力就可以使用這些規則函數來確保程式不會違反企業規則。

雖然具備了資料庫連結能力的程式都可以透過資料庫介面（ODBC, ADO, JDBC等）來使用規則函數，但是這種方式在使用上並不方便，所以需要替程式整理一個方便使用的企業規則庫介面。

而企業規則庫介面所指的是一組爲特定程式語言或工具所開發的函數或是類別。例如爲了讓Java程式能夠更容易的使用規則函數，我們可以將每一個企業規則的所有規則函數都再包裝成一個Java類別，如此Java程式就可以使用這些Java類別來間接使用規則函數。而這一組爲了方便Java程式使用企業規則庫的類別就是一個Java語言的企業規則庫介面。

除此之外，爲了進一步減少修改規則函數的機會，並提升讓使用者自行維護企業規則的可能性，規則參數必需從規則函數中抽離出來。抽離出來的規則參數則儲存在規則參數資料表格之中，而該資料表格則建議採用如圖2所示的設計。

圖 2 規則參數資料表格的設計

五、企業規則庫開發程序

如何成功開發一個採用上節所提架構的企業規則庫？本節的目的即爲了解決這個問題。

圖 3 企業規則庫的開發程序

一個特定資訊系統所需要的企業規則庫，其開發程序如圖3所示，從規則分析開始，接著進行規則庫設計和規則庫實作，然後到最後的規則庫維護，一共是四個步驟。這四個步驟分別說明如后。

（一）企業規則的分析

開發企業規則庫的目的在於滿足資訊

系統的各種企業規則需求。而開發企業規則庫的第一個步驟就是界定企業規則的範圍，並就這個範圍進行企業規則的收集、整理、記錄。這一連串的開發活動就是所謂的企業規則分析。

資訊系統所涉及的作業流程與企業機能，以及資訊系統本身的功能需求與限制等項目共同形成了企業規則的範圍。

而企業規則最常記錄在公司的文件裡，部份則記憶在某些員工的大腦中，但也有可能祇是實作在某個資訊系統裡。這些都是企業規則的可能來源。

開發人員必需從企業規則來源之中，將資訊系統所需要的企業規則挖掘出來。Morgan[6]與von Halle[8]都提出了一些很有建設性的技術和方法，對於企業規則的挖掘相當有幫助，值得參考。

要注意的是，資訊化後的作業程序、人員安排、企業規則等項目並不必然與原來的完全相同。因此除了要挖掘出現有的企業規則，因應資訊化的需要，還必需對這些挖掘出來的企業規則做修訂或增補。

另外，挖掘出來的企業規則必需用一致性且結構化的文字陳述方式予以記錄下來。針對這個問題，歐美的企業規則領域專者建議使用規則樣式（Rule Patterns）來規範並簡單化企業規則的陳述[4,6,8]。然而中文畢竟和英文不相同，中文並不若英文般的結構化。適不適用規則樣式的作法？可不可以直接使用他們所建議的英文樣式？能不能以他們所建議的英文樣式爲基礎來發展出一套中文的規則陳述樣式？這些問題都還是需要予以深入探究和解決的課題。

本文建議一種對企業規則做文字陳述的方法：每位開發人員都先儘量將自己認爲適合的企業規則陳述寫下來，然後舉行一個簡單的討論會議，在會議中說明並檢討每個人所提出來的各種陳述，然後在會議之後爲企業規則決定出一個陳述。

除了用文字來陳述企業規則之外，如果企業規則牽涉了複雜的決策，則建議再使用結構化軟體分析技術[5]裡頭的決策表、決策樹，或是虛擬碼來予以做更清楚的表達。

上述之企業規則庫分析過程中所得到的結果必需予以記錄成分析文件。這份分析文件是接下來的企業規則庫設計活動的行事依據。

（二）企業規則庫的設計與實作

一旦將資訊系統所需要遵守的企業規則都挖掘了出來，並以結構化的方法妥善記錄了下來，那麼就可以開始設計企業規則庫。而設計企業規則庫所要做的事情主要有三項：（1.）決定企業規則要映射（Mapping）成那些規則函數、（2.）設計規則函數、（3.）設計企業規則庫介面。

如何決定企業規則要映射成那些規則

2 目前關連式資料庫管理系統所能夠支援的資料庫完整性確保機制包含有：Domain, Not Null, Unique, Default, Check, Primary Key, Foreign Key, Trigger等

函數？在回答這個問題之前，必需先說明企業規則的各種使用者，以及企業規則將映射成的各種規則函數的特性。

企業規則的使用者有三種。第一種是程式的使用者介面（User Interface），第二種是程式的處理邏輯（Process Logic），第三種是資料庫（Database）。使用者介面對於企業規則的使用，主要是用來進行使用者輸入資料的檢核，例如做身份證字號正確性的檢查，或是判斷住家電話、公司電話、戶藉電話這三個資料輸入欄位是否最少輸入了一項。處理邏輯對於企業規則的使用，主要是用來做資料處理，例如計算員工的薪資、決定銀行貸款的信用評等。資料庫對於企業規則的使用，則主要用來確保資料庫的完整性，而規則函數將使用於關連式資料庫的各種資料庫完整性確保機制[2]之中。

規則函數可以區分成計算型、決策型、資訊型三種。其中，計算型函數可以用來執行一個公司規定的計算法則（Algorithm），例如員工薪資計算函數，以及銀行貸款信用評等函數均爲此種函數。決策型函數則可以用來判斷眞僞或是進行挑選，例如判斷某一客戶是否是會員的函數，以及決定某一員工應該採用那一種基本月薪資的函數均爲此種函數。資訊型函數則用來提供下決策之時所需要使用的重要資訊，例如傳回銀行願意承作客戶的年齡範圍（最小承作年紀與最大承作年紀）的函數，以及傳回員工每年可請事假天數的函數均爲此種函數。

決定企業規則要映射成那些規則函數，就是去辨識出企業規則的使用者，並思考這些使用者的企業規則需求，以及各種規則函數的適用程度、易用程度、可維護程度，然後在權量輕量之後，決定每個企業規則所要實作的規則函數，以及這些函數各自應該負責的功能。

而規則函數的功能一旦予以決定了下來，就可以開始對規則函數進行設計。首先設計規則函數的原型（Prototype），也就是要先決定下來規則函數的名稱、參數，以及傳回值。接著則再依據企業規則的內容、規則函數的功能和原型來設計規則函數的內容（亦即規則函數的處理邏輯）。另規則函數的設計需要予以清楚表達，這方面可以參考結構化軟體分析技術之處理規格（Process Specification）的作法[5]。

另如果所有的規則函數都予以設計完成，亦即企業規則庫已經設計完成，接著便要去思考設計企業規則庫介面的問題。這個設計問題比較單純，基本上祇是將每一個規則函數都包裝成一個介面函數，或是將同一個企業規則所映射的所有規則函數（規則函數群）予以包裝成一個介面類別。而每個介面函數和介面類別的設計都應該予以清楚表示出來，而這方面可以分別參考結構化軟體設計技術[5]和物件導向軟體設計技術[3]所提出的作法。

要注意的是，上述之企業規則庫設計

過程中所得到的結果必需予以記錄成設計文件。這份設計文件是接下來的企業規則庫實作活動的行事依據，也是日後企業規則庫維護的指南。

一旦完成了企業規則庫的設計之後便可以使用設計文件來進行實作作業。先使用資料庫管理系統所提供的資料庫管理工具來撰寫規則函數，再分別使用各種不同的程式開發工具來開發各種不同版本的企業規則庫介面。

（三）企業規則庫的維護

隨著企業規則的改變，或是資訊系統的更動，企業規則庫不免地必需進行修訂或是增補等維護活動。而採行本文所提架構的企業規則庫，其可能的變動情況說明如下。

（1）規則參數的變動：將銀行貸款承作客戶的年齡範圍由原來的18歲至60歲修訂成20歲至65歲，這就是一種規則參數變動的例子。這一種變動一般可以透過簡易的規則參數管理介面讓使用者自己修改，也可以由管理人員使用資料庫管理工具來完成。

（2）規則函數內容的變動：員工薪資計算方式的改變就是一種必需修改規則函數內容的變動。這種變動需要管理人員使用資料庫管理工具來改寫負責該計算的規則函數。

（3）規則函數原型的變動：上述的員工薪資計算方式的改變，如果改變的幅度比較大，不祇是規則函數的內容需要改變，連同函數的原型都必需重新設計，這就是一種規則函數原型的變動。而規則函數原型變動之後，連帶著也要去修改企業規則庫介面裡相對的函數或類別，以及資料庫裡頭使用到規則函數的部份。甚至於如果介面函數或介面類別的原型仍然無法保持不動，那麼程式裡頭有使用到介面函數或介面類別的部份也都必需一一找出來修改。

（4）新增規則或改變原有規則設計的變動：因應經營方式的變動，時而會新增加企業規則，時而會改變原有規則的設計。這時需要處理的維護作業就比較多、比較麻煩。大部份的情況之下，規則參數、規則函數、企業規則庫介面、程式、資料庫等項目不免都要進行修改。

（四）企業規則庫的整合

與資料庫相仿，我們追求企業等級的企業規則庫，如此才能讓企業規則庫的效益最大，管理成本最低。

企業級的企業規則庫由各個企業資訊系統的企業規則庫整理而來。這個整合的企業規則庫是不同資訊系統之間企業規則庫的聯集，其在一樣能夠滿足所有資訊系統的企業規則需求之下，卻可以儘量減少重覆儲存，降低不一致的情形，更有利於管理與維護。

而企業規則庫的整合方式可以區分成事前整合方式（資訊系統規劃）與事後整

合方式（資訊系統整合）兩種，分述如下：

（1）資訊系統規劃：依據企業的願景與策略、科技的現況與趨勢，通盤檢討企業的資訊系統需求，進而衍生出各個資訊系統的企業規則需求，並藉以建立一個企業級的企業規則庫。

（2）資訊系統整合：逐一將新系統的企業規則庫整併進企業級的企業規則庫之中。

六、結論

實務上，開發人員一直致力於將企業規則獨立於程式之外，然而學界直到九〇年代才將企業規則視爲一個研究領域來予以研究，而且企業規則研究領域並沒有很大的進展，仍然有相當多的課題需要予以解決。

本文提出一個以關連式資料庫程序爲基礎的企業規則庫開發策略，企圖扮演在企業規則管理技術還沒有成熟之際的一個相對比較理想的企業規則庫解決方案。

參考文獻

[1]Business Rules Group, “Defining Business Rules: What Are They Really？”, http://www.BusinessRulesGroup.org, 2000.

[2]Date, C.J., An Introduction to Database Systems, Addison-Wesley, 2000.

[3]Fowler, M. and Scott K., UML Distilled: Applying the Standard Object Modeling Language, Addison-Wesley, 1999.

[4]Ross, R.G., Principles of the Business Rule Approach, Addison-Wesley, 2003.

[5]Martin, J. and Mcclure, C., Structured Techniques: The Basis for CASE, Prentice-Hall, 1988.

[6]Morgan,T., Business Rules and Information Systems: Aligning IT with Business Goals, Addison-Wesley, 2002.

[7]Steinke, G.and Nickolette, C., “Business Rules as the Basis of an Organization’s Information Systems”, Industrial Management and Data Systems, Vol. 103, No. 1, 2003, pp. 52-63.

[8] von Halle, B., Business Rules Applied: Building Better Systems Using the Business Rule Approach, Wiley, 2001.