建構開放型系統分析導向之物件資料庫績效評估工作量模型

謝家蘭
政治大學資訊管理學系

許致順
政治大學資訊管理學系

摘要

現今的物件導向資料庫績效評估存在著許多的缺失。*1.*針對特定的應用領域，其測試工作量模型並無法代表使用者的需求。*2.*針對特定的資料模型，主要在衡量資料庫設計之技巧及效能。*3.*測試資料庫綱目變得複雜，測試運算變得繁多，使用者欲自行建立績效評估十分不容易。爲了解決以上的問題，本研究擬使用物件導向系統分析的方法作爲使用者建構工作量模型的工具，經過*1.*工作量需求分析階段。*2.*工作量規則描述階段。*3.*績效評估的測試階段。最後發展出一個用以測試物件導向資料庫的績效評估環境。由於測試之資庫綱目與測試運算均是從使用者之需求著手，將使得測試之工作量對使用者而言，更具一致性、代表性與正確性，測試所得之數據與使用者更具相關性。在物件導向資料庫的應用範圍愈來愈廣泛、資料庫之資料量日益龐大的未來，以使用者需求爲導向之績效評估方法將來會是未來績效評估之發展方向。

關鍵字：物件導向資料庫、績效評估、系統分析方法、物件導向技術

Building A System Analysis Oriented OODBMS Benchmark Workload Model

Jia-Lang Seng
Department of Management Information System
National Cheng Chi University
T.S.Hsu
Department of Management Information System
National Cheng Chi University

ABSTRACT

Current object-oriented database benchmarks have the following drawbacks: (1) They are designed for one specific problem domains. The workload models are useless and meaningless the represent user and application domain. (2) They are designed for one specific data model. Their intent is merely to measure the techniques instead of the design of the databases. (3) As the database schemes become more complex and the test operations get more myriad, it is more difficult for users to implement any of the current benchmarks just based on their specifications. In this research, we propose an alternative workload model method in attempt to resolve the aforementioned issues. This method is system analysis method oriented. We model the workload from the perspective of users' requirements. We divide the workload development process into three phases: (1) workload requirements analysis phase (2) workload requirements specification phase and (3) benchmark experiment phase. Since our benchmark database schema and operations are derived from the user requirements, the workload model becomes more consistent, relevant, representative. Hence, the benchmark results are useful and meaningful. As the diversity of the OODBMS application domains grows and the scale of the OODBMS test suite expands, our new method presented in this paper provides a viable resolution to the inadequacy, ir-representativeness, and in-feasibility of the current benchmark methods.

Keywords: Database Benchmarks, System Analysis Methods, Object-Oriented Database Systems, Object-Oriented Technology

壹、緒論

一、研究背景與動機

資料庫的技術（database technology）正不斷地在發展與改進，以期能持續改善其效能與補足其弱點。近年來，物件導向技術（object-oriented technology, OOT）成爲發展資料軟體系統與資料庫系統方法的一股新的潮流與驅勢，許多物件導向語言因此如雨後春筍般被提出，例如Smalltalk、Actors、Objective C、C++、Java；傳統的關聯式資料庫（relational database management）雖然在效能上有不錯的表現，但是關聯的組織方式資料庫卻不免有其缺失與限制，例如：單純的表格方式並無法自然地表現眞實世界的實體及實體間一般性與特殊性的關係，對於實體間的組合關係亦無法充分顯示，於工程之應用上，其效能表現太差等等。資料庫廠商爲了因應這些趨勢與改善傳統關聯式資料庫之缺點，開始在他們的關聯式資料庫中，加入一些資料結構以支援物件導向的概念，形成降了純物件導向資料庫系統外（object-oriented database management system，OODBMS），所謂的物件關聯式資料庫系統（object-relational database system）。此種方式不僅保留了許多關聯式資料庫系統的特質，且將資料以自然的物件方式予以表達，並應用了許多物件導向技術來表示資料，更由於它的包容性，使得其成爲異質性資料庫系統的共同模式 [12][20]

資料庫系統的效能表現（performance analysis）對於使用者之工作效率有著深遠影響。因此，許多用以衡量與預測資料庫效能與效率之工作量模型（workload model）便孕育而生，一般我們即稱爲資料庫的績效評估（database benchmark）工具。一般的資料庫績效評估規格皆是針對特定的範圍領域建構出某些典型的應用模型，並以合成的工作量（synthetic workload）作爲績效評估之測試依據，若使用者應用資料庫之範圍領域（preset domain）與績效評估工具之問題領域（user domain）存在極大之差異性，結果將使得使用者在實際使用資料庫資料進行交易時，所得到的效能表現可能會與績效評估工具所得到的結果有極大的落差。在此情況下，一般的績效評估工具便無法發揮實際效果；另一方面，資料庫績效評估工具爲了能夠整體且全面地衡量與預測資料庫的眞正效能，將資料庫內的資料結構逐漸演變成十分複雜，測試之運算亦非常的繁複，給予實作之測試者及設計者過重的工作負擔。然而資料庫績效評估工具之可靠性與複雜度是正向關係，一個簡單且易實作的資料庫績效評估規格，可能因其測試資料結構（test data）過於簡單且測試運算（test operation）之廣度與深度不足，易流於形式。無法提供正確且有效的資訊，使得其對於預測資料庫績效之貢獻極微。複雜的資料庫績效評估規格卻因其資料綱目複雜與繁多的測試運算，使得實作上變得十分地困難，且所有之運算結果資訊，對使用者而言，並非全然有用。

本研究擬針對物件導向資料庫的特徵，利用物件導向系統分析方法UML（Unified Modeling Language），建構出開放型系統分析導向物件導向資料庫的工作量模型，發展出一套衡量物件導向資料庫效能之績效評估方法，以使測試之績效評估模型與工作量對於使用者而言具正確性、代表性與一般性。在此模式與方法中，使用者只需使用視覺化的物件導向模型建構工具，來表達其資料庫需求模型，即可根據自己的需求對於特定之物件導向資料庫作績效評估。

二、研究問題陳述

一般資料庫績效評估的方法和規格中（ method and specification ），其測試工作量模型往往使用合成之工作量（ synthetic workload ）作爲測試之依據，並且是對特定之範圍領域建構典型之應用（例如 HyperModel 是針對超文字的應用， OO1 、 OO7 則是針對工程應用問題） [1][9][10] 。若所假設之工作量具普遍性與一般性，則測試所得到的效能指數亦只是在事先定義好的應用領域型態下，所可能得到的系統效能表現，並無法涵蓋所有的資料庫使用需求與可能之工作量。當使用者之應用領域與績效評估規格所假設之應用迥然而異或者工作量與模型之假設差異甚大時，這些績效評估便無法在使用者所應用的領域內，提供有效的以預測與衡量系統之功能。

一般的績效評估規格往往是對特定資料模型（ data model ）資料庫擁有的主要特徵作效能測試。爲求充分了解資料庫管理系統所使用的科技或策略（ technology and/or strategy ），對於資料庫效能會有哪些程度的結果影響，往往從各種的資料庫運算中，挑選運算作測試。然而使用者所關心之議題並非某種資料庫採用何種策略或科技，使用者較關切之焦點在某種資料庫系統對於其於應用上會得到哪種程度之效能和績效。因此，常有 001, 007 , Hyper Model 測試內容與使用者所需相左情形發生。

績效評估規格爲了能夠具廣度與深度地測試資料庫管理系統的各項功能，與因應使用者日益龐大之資料庫資料量與逐漸複雜之應用，所制定的工作量模型之資料庫綱目（ test schema ）與測試運算（ test operation ）有趨於複雜的傾向。但從另一方面看來，這種趨勢將使得資料庫之使用者，欲根據這些績效評估規格自行實作並作測試將變得更加困難。一般的績效評估工作囿於時間與經費，只能針對少數資料庫管理系統作績效評估工作，有些則受限於軟體合約條款，無法將績效評估之結果予以公佈。

綜合上述，目前績效評估的方法和規格主要的缺點分述如下：

1. 它們針對特定的應用領域。其所假設的物件模型與工作量無法代表使用者的需求。
2. 它們針對特定資料模型資料庫之特性作效能測試，主要在衡量所使用的科技或策略之效能表現。無法針對使用者的需求作測試。
3. 它們績效評估之測試工作量模型趨於複雜，運算變得繁複。使用者欲自行實作並予以測試其效能，將需要大量程式設計資源而變得困難。

三、研究目的

本研究的主要目的如下：

1. 發展一個用以建構工作量模型的方法論。此方法論包含必要之步驟與使用之技巧。
2. 利用物件導向系統分析的方法，讓使用者將資料庫使用需求建構成圖形型態且易於了解的物件模型。
3. 將圖形型態物件模型轉換成正式性的文字型態物件模型，表達使用者的測試需求以利後續電腦內部的編譯、運作與儲存。

貳、UML物件導向之系統分析方法

1994 年 Rational 軟體公司的 Grady Booch 與 Rumbaugh 開始致力於合併 Booch 與 OMT 的方法，取此二者長處的

UML（Unified Modeling Language）方法即發軔誕生。至 1995 年秋季，Jacobson 亦加入統一物件導向系統分析與設計的工作，將 OOSE 的方法予以合併採用 [5] [17][28]30] 。1997 年 11 月，OMG 的理事會通過將加入 UML 的規格於其物件管理結構（OMA，Object Management Architecture）的決議，使得物件導向系統分析與設計能夠有一致的語言來發展軟體系統。

一、觀點（Concept）

觀點並非圖形，而是由一些圖形所構成的抽象化概念，藉由定義顯示系統不同觀點，描畫出整個系統的情況。UML 包含了以下幾種的觀點：

1. 使用者－案例觀點（Use Case）：觀點是由外部使用者所察覺的系統功能。
2. 邏輯觀點（Logic）：此觀點是以系統靜態與動態行爲的方式來顯示系統內部如何設計各功能。
3. 元件觀點（Component）：此觀點顯示程式碼元件的組織方式。
4. 等時性觀點（Sequence）：此觀點顯示了系統等時性。
5. 採用觀點（Depolyment）：此觀點顯示系統採用何種結構與設備。

二、圖形（Graph）

圖形是用以描述觀點內容的圖案，UML 提供了九種不同的圖形以描述系統的不同觀點。包括使用案例圖（use-case diagram）、類別圖（class diagram）、物件圖（object diagram）、狀態遞移圖（state diagram）、順序圖（sequence diagram）、合作圖（collaboration diagram）、活動圖（activity diagram）、元件圖（component diagram）及採用圖（deployment diagram）。

三、模型元素（Model）

圖形內所使用的概念即是模型元素，模型元素代表一般物件導向的概念例如類別、物件與訊息，模型元素可以使用在不同的圖形裡，但無論在哪一種圖形裡均有同樣的意義與符號。

四、一般的機制（Rule）

一般的機制對於模型元素提供額外的註解、資訊與語意，當使用一般的模型元素無法予以清楚表達某概念時，可以使用這些機制以表達。

參、績效評估工作量模型

一、資料庫工作量模型

電腦系統可以視爲被利用在處理使用者需求的軟體與硬體集合，於某一段時間內，使用者藉由輸入程式、資料或者命令以送出其處理要求，這些輸入之資訊即被稱爲一個集合用語-工作量。系統的效能指數必須在給定說明在多少數量之工作量模式下運作方有意義。更進一步的說，對於需要作不同系統或設定之效能比較工作，例如調整或者設計系統等活動，只有在所有欲比較的系統間，工作量模式相同的情況下，系統效能指數才有意義。

對於用以測試系統效能指數之程式、資料或者命令之集合所建構的模型系統我們即稱爲工作量模型（workload model），測試工作量模型可以被分成不同層次與類別。以層次來分，可以大致分爲三個層次 [18]：

1. 實體的層次：這是第一個層次，工作量模型是以消耗系統之軟體或者硬體資源爲基礎，例如基本的工作量模型元件是以消耗中央處理器（CPU）的時間、

所執行的指示數量、系統所需的主記憶空間等爲其係數，這個層次的模型以資源爲導向（ resource-oriented ）爲其主要特徵。

2. 虛擬的層次：第二個層次考慮系統的邏輯資源，例如工作量模型的基本元件乃是以高階語言的程式碼行數、檔案或者資料庫之存取數目或者所需的虛擬記憶體空間爲其特徵。
3. 功能的層次：這個層次之工作量是以它所組成之應用爲主，幾個功能相似的程式欲作測試，例如存貨控制系統、排序演算法、編譯器之效能測試等。工作量的功能必須與系統相互獨立，此系統大多用於採購研究，因採購須要考慮不同系統或者不同設定之效能比較。

測試之工作量若依所產生的方法分類可以分爲三類 [18] ：

1. 實際之工作量：此工作量是在一定時間內，由原始要處理的程式與資料所構成的。因此，這個種類的工作量乃是在某段衡量的時間內，實際系統所處理的工作量。
2. 混合之工作量：由實際之工作量元件與刻意建構之元件所混合構成之工作量稱爲混合之測試工作量 。
3. 合成之工作量：未使用任何實際工作量元件稱爲合成之工作量。

二、物件導向資料庫之績效評估方法

績效評估系統必須具簡單性、可調整性、可攜性、可重覆性、接受性及相關性 [19] 。有時這些目標是相互衝突的，例如在可攜性與相關性之間，往往必須作某一程度的取捨，而可調整性與簡單性的兩個目標間亦存有此種關係。

在物件導向資料庫管理系統之績效評估方面， OO1 是衡量資料庫於工程應用上（ 如 CAD/CAM /CASE ）效能表現的績效評估規格，由於它具簡單性的原則且是第一個針對物件導向資料庫所擬定出來之規格，使得它在物件導向資料庫的績效評估方面，成爲了一個實際的標準 ； HyperModel 模型建構在超文字的應用模型上，其主要目的，是爲了更廣泛地測試工程應用上資料庫的效能表現，它比 OO1 包含更豐富的資料型態與範圍更廣之運算； OO7 績效評估的對象如同 OO1 ，在衡量物件導向資料庫管理系統在各種工程應用上之效能，它是一個對於物件導向資料庫管理系統作全面測試的績效評估標準，因此相較於其它績效評估的標準， OO7 擁有較複雜的綱目（ schema ）與較繁多的運算（ operations ）； BUCKY 則是用於衡量物件關聯式資料庫之效能，主要是針對物件關聯式資料庫所附加的「物件」特徵，作各種查詢（ query ）運算測試，包括繼承（ inheritance ）、物件間之參考（ inter-object reference ）、集合值之屬性（ set-valued Attributes ）、物件之方法（ methods of objects ）、抽象資料型態之屬性與方法（ ADT of attributes and methods ）。

三、比較物件導向資料庫績效評估方法規格和工作量模型

以下表 1 是針對上述討論的幾種績效評估作一整理與比較：

肆、研究方法—開放型系統分析方法導向之工作量模型

本研究方法是提出一項開放型，以系統分導向的物件導向資料庫績效評估方法。以使用者的需求著手，使用物件導向系統分析方法來建構使用者之工作量模型

表1：績效評估規格之比較(本研究提出)

	假設的領域	測試運算的種類	使用繼承的特性	物件間的關聯（參考）
OO1	工程設計應用	查詢、尋訪與插入物件	否	是
HyperModel	超文字的應用	查詢、尋訪與更新物件內容	否	是
OO7	工程設計應用	廣泛的查詢、尋訪與更新	是	是
BUCKY	一般（大學資料庫）	各種的查詢測試	是	是

續表1

	使用結合（複雜物件）的特性	使用集合（set）的特性	使用類別的類別（metaclass）特性	物件方法的使用
OO1	否	是	否	是
HyperModel	否	是	否	是
OO7	是	是	否	是
BUCKY	否	是	否	是

[35][36]。我們將發展此資料庫績效評估模型分成二個階段。第一階段是「工作量需求分析階段」，此階段的主要目的是建構一個簡單、易於明瞭的圖形型態工作量模型；第二個階段則是「工作量規格描述階段」，主要是將系統分析階段所建構出的圖形型態工作量模型對映文字型態工作量模型。

第一個步驟的主要工作，在於選擇一個適當的系統分析方法，並經由延伸（extension）和特別化（specialization），能夠建構出精確、合適與代表使用者需求的工作量模型，此將是影響一個以使用者需求爲導向之績效評估模型關鍵因素。前一章所介紹的幾個物件導向績效評估之工作量模型，因假設的應用領域殊異且著重的焦點不同，資料庫綱目與測試運算因此迥然而異。若能使用物件導向系統分析方法，將使用者之需求以精密、簡潔、易於了解的模型加以表達。本研究擬利用UML於物件導向系統分析階段所提供的視覺化圖形符號。加以延伸及特別化，另自行設計本工作量模式的文字型態規格與語法。

第二個階段是將圖形型態工作量模型對映至文字型態的規格和語法，以產生工作量模型的規格描述，這些工作量規格描述將是建構績效評估應用程式之依據。爲了兼顧系統運作之便利與使用者可讀性的問題，我們所發展出的語言語法是一種概念層次類似自然語言的規格語言。

同時，在本文中將以概念一型架構規格語言（Conceptual Framework）探討如何進行實驗設計和執行績效評估。我們將此部份分成三個模組元件，第一個是資料庫產生器元件（database generator），另一個是交易產生器元件（transaction generator），最後則是測試驅動器元件（test driver）。這三個元件各司不同之職責與發揮不同之功能。資料庫產生器針對工作量模型裡的資料模型，藉以產生測試之資料庫；交易產生器則是針對工作量模型所描繪出的單一交易圖形與複合交易圖形，用以產生不同的測試運算；測試驅動器則是根據控制模型中的交易順序與次數，使這個物件導向績效評估系統運作起

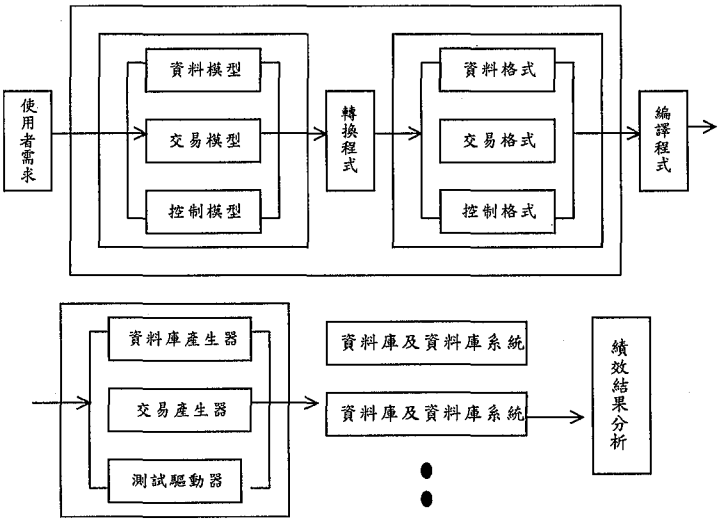

圖1：本研究的研究模式圖

來，產生資料庫之測試評估數據。本研究的整體研究模式如圖 1 所示。

一、工作量需求分析階段

工作量需求分析階段的主要目的是爲了建構一個能夠代表使用者應用物件導向資料庫的工作量模型。

我們將工作量模型依不同的觀點區分成三個部份，分別爲資料模型、交易模型與控制模型。資料模型主要是用來描述工作量模型裡的資料庫結構、大小與特質，是系統靜態的觀點。使用類別圖與類別關係圖來描述不同型態與類別間的關係及類別間的結構。交易模型主要是用來描述工作量模型裡交易處理的流程與處理邏輯，是系統動態的觀點。使用物件狀態圖與物件合作圖來描述關於系統執行期間，所發生的重要事件，例如各類別的案例與案例間溝通的訊息與事件。控制模型則是描述各筆交易執行的順序與交易次數。由於使用簡單的表列即可表達交易的順序、種類與次數限制，因爲我們並沒有使用圖形來表達控制模型，系統可使用任何表單之方式讓使用者輸入。圖 2 是工作量模型結構圖。

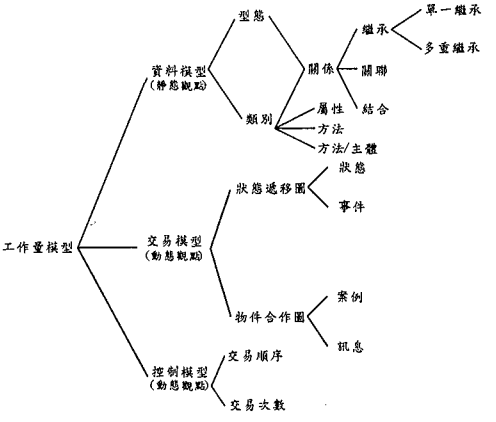

圖2：是工作量模型結構圖。

圖3：型態的分類

(一)資料模型

資料模型主要是以系統靜態、結構與資料的觀點，來描述工作量模型裡之交易的資料。資料模型可包含型態（ type ）與類別（ class ）的概念。型態用以描述值（ value ），因此值是它的案例，值沒有識別值（ identity ）可識別其差異性；類別用以描述物件，因此物件是它的案例，物件有辨識值以識別其差異性，所有擁有相似行爲與屬性的物件集合即形成類別，類別的定義是物件導向綱目的重心所在。不同的型態與類別均可使用關係來表達彼此間邏輯上的關聯性，關係大致可區分成繼承（ inheritance ）、關聯（ association ）與組合（ aggregation ）。以下分別介紹如何使用圖形方式來表達型態、類別與類別間的關係：

1.型態

型態是用以描述值（ value ），因此值即是它產生的實際案例，值沒有識別值（ identity ）可識別其差異性。例如日期或坐標都是一種型態。型態的案例是值（ value ），值存在物件的屬性或者運算的係數裡，因此我們在介紹類別圖的結構前，先定義型態的種類。我們將型態又分成數類，如圖 3 所示。

基本的型態種類往往與程式語言的環境、實作所使用的語言或資料庫平台有很大關係。我們使用一般的分類方式來表示基本的型態種類，包括 integer 、 real 、 boolean 與 string ，分別表示整數、實數、布林數與字串；物件的識別（ OID ）是參考其它實體物件的符號。因此，我們將物件的識別加入可能的型態集合裡。當資料型態是參考其它類別物件的指標時，我們使用物件類別的名稱來表示，然而類別型態的值實際上是以物件的識別（ OID ）來儲存與運作；結構化的型態是基本型態值的集合，我們將其分成 set 、 bag 、 list 與 tuple 。 set 表達未排序、不重複之相同資料型態的集合； bag 則表達未排序、可重複之相同資料型態的集合； list 表達排序且可重複之相同資料型態的集合； tuple 則用以表達排序、可重複且可爲不同資料型態的集合； array 則用以表達排序、有編號及不重覆之相同資料型。

2.類別

類別圖主要如圖 4 所示，類別圖分成三個部份。第一個部份是類別的名稱，第二個部份爲類別所擁有的屬性，屬性依種類與複雜程度可以區分成數種；第三個部份則是類別的方法，方法可能是提供其它物件的服務或者對於自身的運算。

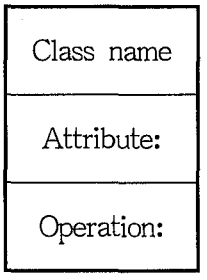

圖4：類別圖的基本結構

以下分別針對屬性與方法兩個部份作說明：

(1)屬性與方法的表達法

屬性與運算的表達語法，如以下所示。屬性之表達法乃依序表達屬性的名稱、屬性的型態與預設值；運算之表達法則依序表達運算的名稱、運算的係數型態與運算的傳回型態。

attribute_name: type = default_value
operation_name ' ('argument-list') ': return-type

我們針對不同屬性的型態與依其複雜程度又分別幾種：

a. 簡單屬性（ simple attribute ）：簡單屬性乃是指屬性型態爲系統內建的基本型態，如前所述因基本型態的種類與系統環境與使用的語言有關係。我們使用系統內定的資料型態爲整數、實數、布林數與字串，在屬性的型態項（ type ）分別以關鍵字 integer 、 real 、 boolean 與 string 表示。

b. 參考屬性（ reference attribute ）：參考屬性的資料型態是屬性型態爲物件識別的屬性，參考屬性被用於代表物件間存在的關係。參考屬性很類似程式語言裡指標（ pointer ）的概念，亦與關聯式資料庫系統中的外鍵（ foreign key ）有點雷同，然而卻存在些許的差異性。首先，參考的屬性不能夠被破壞，而指標卻是可以的，當被參考的物件被刪除時，參考的指標將自動變成無效；再者，參考的屬性與使用者可見的值並不相關，而外鍵的值卻是相同的。被參考的物件之內含值可能在不同情況下會有改變，但是指向它的參考屬性值卻依舊指向此物件，實際上參考的屬性是以物件的識別來儲存。我們將屬性的型態用物件類別的名稱表示這是一個參考的屬性。例如一個學生類別，擁有屬性爲主修科目，主修科目是資料庫另一類別的資料，因此可使用參考的屬性予以表達，以下即是類別爲學生所擁有參考屬性主修的表達法：

major : department

c. 集合屬性（ collection attribute ）：集合屬性的資料型態爲結構化的型態。集合的屬性用於表達屬性爲某種資料型態的集合。如前所述，我們使用了包括 set 、 bag 、 list 、 array 與 tuple 等關鍵字表達不同的集合概念。例如我們欲表達員工資料庫內，其屬性爲兒女姓名的欄位，資料型態爲字串的集合，我們可以使用以下的方式宣告：

children: set [string]

d. 衍生屬性（ derived attribute ）：屬性可以不以明確的值儲存，而是以定義程序的方式來儲存，當欲擷取或者指定某一個屬性的值時，被指定的程序即被執行。例如某員工的薪資是經由呼叫 salary 方法計算而來，即可使用以下的方式定義：

salary_value : integer = salary ( )

3. 型態與類別間的關係

型態與類別間的關係代表類別或型態間存在著某些邏輯上的連接關係，型態與類別間的關係分成三種，分別爲繼承、關聯與組合。

繼承關係能夠表達型態或類別間特殊性 / 一般性的概念；擁有繼承關係的型態

或類別共享著相同的屬性與方法，繼承可能是單一繼承（ single inheritance ）或爲多重繼承（ multiple inheritance ），單一繼承表示子類別的父類別只能有一種類別，多重繼承則允許擁有多個類別的父類別。在型態階層裡，繼承代表著 is_a 的關係，表示著父型態 / 子型態（ supertyping/subtyping ）的關係；在類別階層的繼承關係通常顯示多項的特徵，包括附加（ addition ）、再定義（ redefinition ）、限制（ restriction ），附加表子類別加入其它額外的特徵；再定義表示被繼承的特徵重新被定義；限制表示子類別繼承部份的屬性。類別間繼承的關係其符號如圖 5 所示。

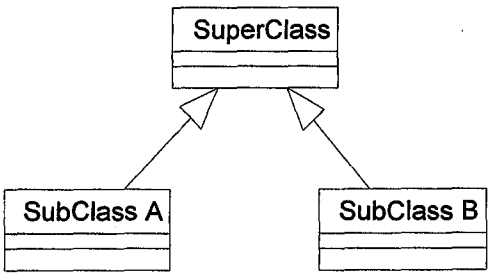

圖5：繼承關係的表達法

關聯則指出型態或類別間彼此邏輯上的關係，因此某一類別的改變勢必對另一個類別產生影響。類別間的關聯主要爲朋友（ friend ）關聯、使用（ using ）關聯及實例化關聯（ instantiate ）。朋友關聯表示一個類別不受封裝的限制，可以存取另一類別的資料結構；使用關聯表示一個類別使用到另一類別的服務；實例化則表示一個類別是另一個類別實例化的結果。類別間的關聯關係以圖 6 所示。

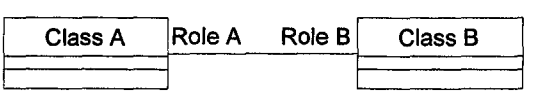

圖6：類別間關聯的表達法

組合（ aggregation ）表達著整體-部份（ whole-parts ）的概念，高階的類別可能封裝低階的構成類別。因此低階的類別在封裝的類別之外，是不可見的，封裝的類別提供介面以供內外界溝通之用。圖 7 是表示組合概念的圖形，空心的三角形一端代表整體的類別，另一端代表部份的類別。

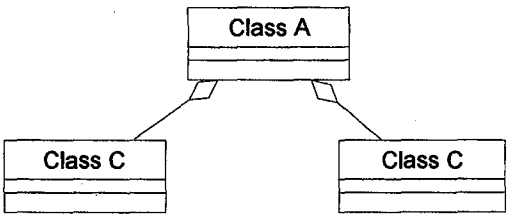

圖7：結合關係的表達

(二)交易模型

交易模型主要是從系統動態、行爲與時間上的觀點來描述整個工作量模型，主要在表達系統交易的流程與處理邏輯。由圖 2 工作量模型的結構圖看來，交易主要分成二種概念，第一種爲單一交易模型，是系統執行時與案例（ instance ）相關的行爲，包括物件從創造至毀滅的生命週期，會有許多不同的狀態（ state ），許多狀態間因事件之發生所引起的遞移關係。我們使用物件狀態遞移圖來表示。由於僅牽涉一種類別物件的交易，因此我們稱爲單一交易模型；另一種稱爲複合交易處理模型，與物件行爲相關聯，由許多物件間相互溝通訊息與彼此間之互動行爲來達成某種交易行爲，我們使用物件合作圖（ collaboration diagram ）來表示。

1. 單一交易模型

單一交易模型是使用物件狀態遞移圖來表示。物件的狀態遞移圖乃是表示物件在它的生命週期內，接受到刺激後，所引

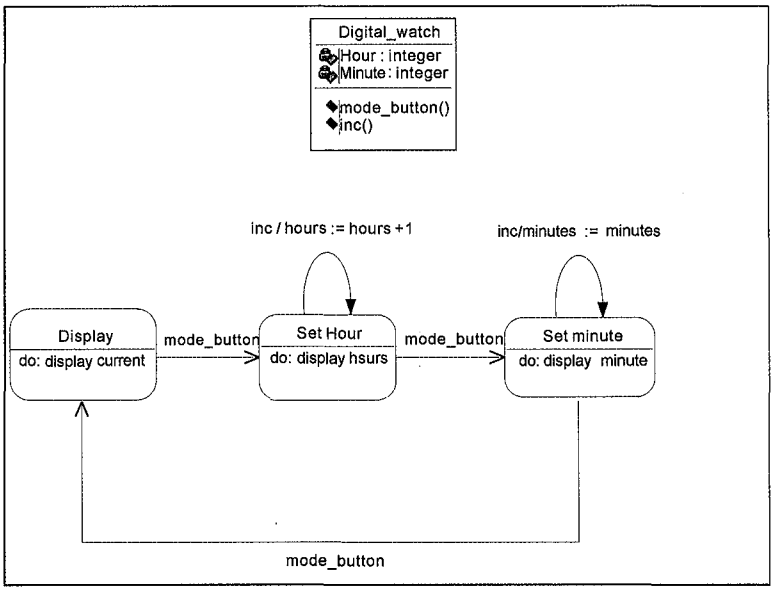

圖8：狀態遞移圖的例子

起的一連串狀態的改變。物件的狀態圖又可分成狀態（ state ）、事件（ event ）與遞移（ transition ）來作說明：

(1)狀態

狀態是物件屬性值的抽象化概念，所有的屬性值集合成狀態，並且影響到物件的整體行爲，狀態則影響到物件對於輸入事件的回應。物件的狀態是以圓角的長方形予以表示，如圖 8 所示。狀態分成三個部份，狀態的名稱、狀態所使用到的變數及內部使用到的活動等。內部所使用到活動以下的語法使用：

event-name argument-list '/' action-expression

上述的語法用以表達處於某種狀態下，發生某些事件，所將採取的行動，event_name 、 argument-list 、 action-expression 分別表示事件的名稱、事件的係數與將採取的行動。

以下幾種的特殊事件，則使用特殊的保留字：

'entry' '/' action-expression 進入某一狀態所要從事的活動。

'exit' '/' action-expression 離開某一狀態所要從事的活動。

'do' '/' action-expression 在某一狀態內，所要從事的活動。

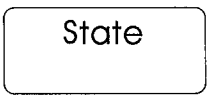

圖9：狀態的表示法

(2)事件

在狀態遞移圖中，事件會引起物件狀態的轉移，事件可能是(1)滿足某些情況(2)從一個物件接受到明確的訊息(3)從一個物件接受到呼叫某些函數(4)經過某段特定的時間。事件先表達其事件名稱再加上其係數，如下的語法所示：

event-signature := event-name ' (' comma-separated-parameter-list ') '

係數則以以下的格式：

parameter-name ':' type-expression

(3)狀態的遞移

狀態的遞移是兩個狀態間的關係，指

出當某些特定情況滿足時，第一種狀態即將進入第二種狀態，且從事某些特定的運算。事件的係數可以被狀態遞移時的行動所使用。在狀態遞圖中，狀態的遞移是以連接兩狀態的實心箭頭所表示，並且在旁邊標上遞移字串，遞移字串如下所示：

event-signature '[' guard-condition ']' '/' action-expression '(' send-clause

event-signature : = event-name '(' parameter ',' '.....') '

event-signature 是用來表達造成狀態遞移的事件名稱與係數。

guard-condition 是以事件、屬性與其它物件之連接所表示的布林表示式。

action-expression 是當遞移即將發生時，所必須執行的程序性表示式。

send-clause 是表示狀態遞移時所傳送的訊息，包括兩部份分別爲目的表達式（ destination-expression ）與事件名稱（ event name ），有以下的形式：

destination-expression '.' destination-event-name '(' argument '.' ....') '

以下是狀態遞移字串的例子，表示當使用者按下滑鼠右鍵的事件發生時，若坐標落於視窗內，將送出視窗高亮度的事件。

right-mouse-down (location) [location in window] / object :=pick-object (location) ^object.highlight ( )

2.複合交易模型

複合交易模型是使用物件合作圖來表達交易概念。物件合作圖主要爲了描述物件間的交互作用與訊息傳遞關係，行爲乃是經由一群物件交互作用並溝通訊息所致，因此物件合作圖可以用以表達交易牽涉到資料與物件函式之呼叫順序。

物件合作圖分成兩個部份，第一個部份是有哪些物件在合作圖內扮演角色，這是物件合作圖靜態的部份；第二個部份則是物件間爲了達成某些特定的目的，彼此間相互溝通的訊息順序等，這是物件合作圖動態的部份。

物件合作圖的物件部份，我們可以以類別圖來表達哪些物件參與物件合作圖，至於物件的屬性與方法，靜態模型中已明確表達。因此，只要說明其名稱即可。物件合作圖的另一個部份，是表達物件間的交互作用，物件藉由相互交換訊息，以達成某些目的。訊息包含訊號（ signal ）與呼叫（ call ）或者更不明確的條件（ condition ）與時間事件。訊息主要是在物件的連接關係旁加一箭頭，連接代表由來源物件傳遞訊息至目標物件。因此箭頭的方向指到目標物件上。其訊息格式如下所示：

predecessor guard-clause sequence-expression return-value := signature

guard-condition := '[' condition-clause ']'

signature := message-name '(' argument-list ')'

predecessor ：表示訊息執行的同步性問題，在現在的訊息送出之前，所有較小序號的的訊息傳遞必須先予以執行。

condition-clause ：代表訊息引發的條件爲何，條件的寫法可以使用一般程式語言的寫法，例如 [X>Y]，代表 X 大於 Y 時才執行。

sequence-expression ：訊息執行的順序。

message-name ：訊息名稱代表目標物件所引起的事件。訊息可以不同的方式來實作，例如函數的呼叫，如果是以函數方式來實作，訊息名稱即是運算的名稱且運作必須是目標物件的方法或者由它來繼承。

return-value ：這是在物件互動關係執行完後，所傳回的係數。若訊息沒有回傳值，則此項內容可以是空的。

argument-list ：在括號內以逗點分開的係數，若沒有傳入的係數，括號以內的內容則是空的。

二、工作量規格描述階段

第二個階段的工作是將工作量分析階段所繪製的圖形型態物件模型對映至電腦可以處理與儲存的文字型態物件模型。我們使用類似綱目定義語言（ schema definition language ）的方式來對映工作量模型的靜態觀點－資料模型。再利用一些程序與運算的表達方式來對映物件的動態觀點－交易模型與控制模型。爲了能夠更嚴謹地定義我們的工作量模型語言，且可讓下一步驟的語法分析器（ parser ）作分析，我們使用了 BNF （ Backus Naur Form ）的形式來定義我們的規格描述語言。 BNF 的符號是採用替代規則（ substitute rule ）或生產規則（ production rule ）的方式，每一個符號皆擁有一個生產規則，符號可能是個句子、變數名稱或者分號。以下是績效評估的規格描述語言之語法定義。在接下來的部份則依序說明如何使用規格描述語言，來描述工作量模型的資料模型、交易模型與控制模型。

```
<benchmark specification>
::= DEFINE BENCHMARK FOR <benchmark name><workload specification>END BENCHMARK
<benchmark name>
::= <identifier>
<workload specification>
::=DEFINE WORKLOAD FOR <workload number> <workload name><data specification><transaction
specification><control specification>END WORKLOAD
<workload number>
::= <integer>
<workload name>
::= <identifier>
<data specification>
::= DEFINE DATA SPECIFICATION<class definition>...END DATA SPECIFICATION
<transaction specification>
::=DEFINE TRANSACTION SPECIFICATION<singular transaction specification> ... |<compound trans-
action specification>...END TRANSACTION SPECIFICATION

以下則是一些基本的定義：
<character> ::= <digit> | <letter> | <special character>
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<letter> ::=<lower case letter> | <upper case letter>
<lower case letter> ::= a | b | c | d | e | f | g | h | i | j | k | l | m |n
|o | p | q | r | s | t | u | v | w | x | y | z
<upper case letter> ::= A| B | C | D | E | F | G | H | I |J | K | L | M |O |
 P | Q | R | S | T | U | V | W | X | Y |Z
<special character> ::= ! | @ | # | $ | ^ | & | *
<data type> ::= <basic data type> | <user-defined data type>
                | <structured data type expression>
<basic data type> ::= INTEGER | REAL | BOOLEAN | STRING | TIME | DATE | DECIMAL
| MONEY
<user-defined data type> ::= <upper case letter>[{<,upper case letter>}]
<structured data type expression> ::= <structured data type> ( <basic data type> )
<structured data type> ::= SET | BAG | LIST | TUPLE | ARRAY
<compare operator> ::= <>| = | < | > | <= | >=
<string> ::= <character>...
<identifier> ::= <upper case letter> [ { [_] { <upper case letter> | <digit> } } ]
<numeric literal> ::= <exact numeric literal> | <approximate numeric literal> |
                      [ + | - ] <unsigned integer> [ . <unsigned integer> ] |
                      [ + | - ] <unsigned integer>. |
                      [+ | - ] .unsigned integer>
<unsigned integer> ::= <digit>...
<approximate numeric literal ::= <mantissa> E <exponent>
<mantissa> ::= <exact numeric literal>
<exponent> ::= [ + | - ] <digit> ...
```

(一)靜態觀點的資料模型

如前所述，資料模型主要是描述資料庫內包含哪些物件類別、類別間的關係與類別的結構等。以下我們將依序介紹如何使用規格描述語言之語法來描述類別與類別間的關係。

1.類別的表示法

類別是靜態物件模型裡最重要的概念，規格描述語言的類別描述擁有以下幾個元件：

- 資料筆數（ number of rows ）：指定此類別的物件個數
- 父類別（ superclasses ）：定義類別之所有直接父類別。
- 屬性（ attributes ）：不同屬性值構成物件的狀態。
- 運算（ operations ）：存取與轉換物件的狀態。
- 鍵值（ key ）與單一性（ uniqueness ）的限制。

以上有些元件並非必要，爲了能夠清楚辨別類別定義的每個元件（例如屬性、方法、限制式等），每個元件的名稱都必須是具唯一性，不可有相同名稱出現的情況。不同類別的元件則允許擁有相同名稱，但爲了避免混淆，元件名稱前方要加上類別的名稱與逗點，以下則是物件類別定義的語法：

```
<class definition>
::= DEFINE OBJECT CLASS FOR<class name>NUMBER_OF_ROWS <number of rows>[SUPER-
CLASSES_ARE <superclass name>...]ATTRIBUTES <attribute definition>...[OPERATIONS <operation
definition>...][KEY <key definitions>]END OBJECT CLASS
<class name>
::= <identifier>
<superclass name>
::= <identifier>
<attribute definition>
::= <attribute name> : <data type>
<attribute name>
::= <identifier>
<operation definition>
::=<operation name> ( operation augment ) :<return value data type>
<operation name>
::= <identifier>
<operation augment>
::= <data type> [{ , <data type>}]
<return value data type>
::= <data type>
<constraint definition>
::= <attribute name> <compare operator > <compare target>
<compare target>
::= <attribute name> | <numeric literal>
<key definition>
::= ( <attribute name>[ { , <attribute name> } ] )
```

2.類別間的關係

在圖形型態的資料模型裡，我們以空心的箭頭表示類別間繼承的關係，因此可以將許多的類別表示成類別階層圖 ，子類別藉由定義 SUPERCLASS_ARE 元件項來指出其父類別的名稱。

類別除了上述的繼承關係，類別間的語意連接關係在語言間並無法直接支援，因此必須轉換成文字型態的另一種概念。這有點類似資料庫設計中，將實體關係圖轉換成關聯式綱目，使用此種的對映方式，類別間的關係概念可藉由加入用以描述關係的屬性或者引入新的關係來表達。換言之，即是使用額外的屬性或者引入額外的類別來將關係的概念引入模型內。在物件導向模型裡，物件間的關係常使用型

態爲物件值的屬性，由一個物件指向一個或多個物件的方式來予以表達。一般而言，關係的表達僅使用一個指標即可表達。然而有時，屬於此關係的兩種物件類別皆儲存指標更能清楚表達此種關係，如此更利於雙向的尋訪此物件關係，缺點則爲會增加額外的管理成本。

(二)動態觀點的交易模型與控制模型

工作量模型的動態觀點主要包含交易模型與控制模型，交易模型又可分成單一交易模型與複合交易模型。在圖形型態的工作量模型裡，分別使用狀態遞移圖與物件合作圖來表達這些概念。以下則說明如何使用文字來描述工作量模型的動態觀點-交易模型與控制模型。

1.單一交易模型

單一交易模型使用物件狀態遞移圖來描述其交易流程。狀態遞移圖可以藉由物件的運算對映至文字型態的語言。以運算爲基礎的設計，主要是指定運算在狀態遞移階段的運算責任。因爲運算有一個條件可以決定行動是否要予執行，當條件成立時，運算才予以執行。由以上的描述，運算分成兩個部份，運算的條件部份表達狀態轉移圖的條件。此條件檢查運算是否必須執行，通常這必須檢視物件的狀態；運算的行動部份是在條件成立時，藉由適當的更新屬性值以改變物件的狀態。以下則是使用文字定義單一交易的語法：

```
<singular transaction definition>
::=DEFINE SINGULAR TRANSACTION FOR < transaction number><class name>
<singular transaction entry>...END SINGULAR TRANSACTION
<singular transaction entry>
::=STATE FROM <state name><state transition entry>STATE TO<state name> <transaction number>
=<integer>
<state name>
::= <identifier>
<state transition entry>
::= <event signature> [ <guard condition> ] / <action expression> ^ <send-clause><class name>
::= <identifier>
<event signature>
::= <event name> ( <parameter> ... )
<guard condition>
::= <attribute name> <compare operator> <numeric literal>|<string>
<action expression>
::= <operation name> ( <operation argument> )
<operation argument>
:: = <data type> [{ , <data type>}]
<send clause>
::= <object name>.<event signature>
<real value augment>
::= <string> | <numeric literal>[ { , <string>| ,<numeric literal>}]
```

<transaction number> 代表此交易在單一交易模型的號碼，<class name> 是單一交易運作的類別名稱，<singular transaction entry> 則是每個狀態遞移的資料，包括狀態遞移的來源與目的狀態與遞移的字串。

2.複合交易模型

複合交易模型使用物件合作圖來表達交易的邏輯與各物件間的函式呼叫。物件合作圖包含兩部份，表達有哪些物件及彼此間的交互作用關係。我們使用以下的語法來對映物件合作圖：

```
<compound transaction definition>
::= DEFINE COMPOUND TRANSACTION <transaction number> <compound transaction name>
<message entry>...END COMPOUND TRANSACTION
<transaction number>
::= <integer>
<transaction name>
::= <identifier>
<message entry>
::= NUMBER <sequence number>[CONDITION <condition expression>]MESSAGE FROM CLASS
<class name>[OBJECT <object name>]MESSAGE<message string>MESSAGE TO CLASS <class name>
[OBJECT <object name>]
<sequence number>
::= <unsigned integer>
<condition expression>
::=<condition>
<class name>
::= <identifier>
<object name>
::= <identifier>
<message string> ::= <operation name> ( <operation augment> )
```

<transaction number> 表示訊息的順序； <compound transaction name> 是表示我們命名之複合交易名稱；來源物件與目標物件都必須說明其物件類別，而物件名稱則是選擇性； <condition expression> 表示訊息傳遞的條件， <message string> 則是表達物件合作圖所傳遞的訊息字串。

***3.*控制模型**

控制模型並沒有圖形來表示，我們使用以下簡單的語法來表達各交易處理的順序與次數。

```
<control specification>
::= DEFINE CONTROL SPECIFICATION<transaction entry>...END CONTROL SPECIFICATION
<transaction entry> ::=<transaction type> <transaction number>TIMES <repetition number>
<transaction type>
::= SINGULAR TRANSACTION | COMPOUND TRANSACTION
<transaction name>
 ::= <identifier>
<repetition number> ::= <integer>
```

其中 <transaction type> 代表交易的種類； <transaction number> 代表各種交易的號碼； <repetition number> 則是交易的次數。

三、績效評估測試階段—觀念架構

本節僅就績效評估測試階段，作一概念性架構的探討和提出。利用工作量模型的規格描述語言作爲建立績效評估程式的基礎。再使用語法分析器（ parser ）來分析文字型態工作量模型，並產生實際可執行的程式碼。本研究將相關文獻予以整理與歸納，而將績效評估應用程式分成三個元件，分別爲資料庫產生器（ database generator ）、交易產生器（ transaction generator ）與測試驅動器（ test driver ）。資料庫產生器根據使用者輸入的資料模型，對映成運作平台上的綱目定義語言形式，以產生資料庫的資料綱目，再根據資料庫綱目產生實際運作的資料庫資料。交易產生器則是根據使用者所輸入的交易模型，對映並轉換成各種測試運算的函式呼叫程式碼。測試驅動器則是績效評估程式之核心，主要的功能是有次序性地驅動所有交易運算，交易運算之執行順序代表使用者應用資料庫的邏輯順序。以下則針對績效評估應用程式的三個元件作概念性說明：

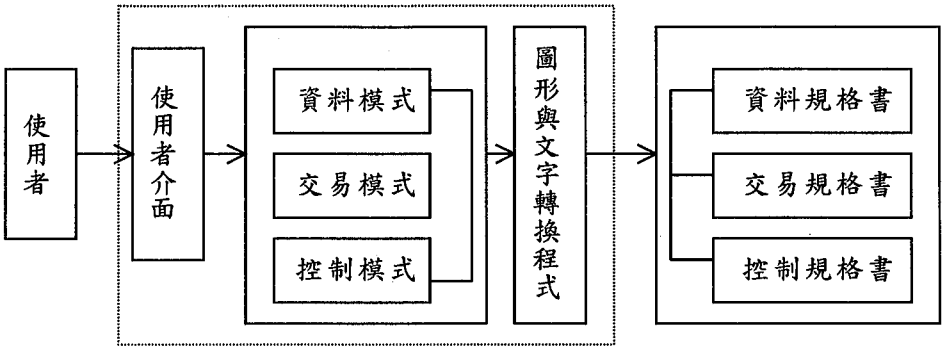

圖10：實作系統整體架構圖

㈠資料庫產生器

資料庫產生器的主要目的，是根據資料模型產生測試運算處理的資料。工作量模型的資料庫綱目之複雜度與物件導向資料庫之效能息息相關。因此若能正確地表達資料庫的綱目與運作資料庫的大小，將是工作量模型能否具代表性與正確性的主要因素。

系統發展者在系統實作階段將靜態文字型態的物件模型對映至資料庫產生器的綱目定義語言，以開始產生運作之資料庫。綱目定義語言是依實作平台與物件導向資料庫種類，其語法與支援的物件概念亦有所差異。

㈡交易產生器

交易產生器主要是產生測試資料庫運算的相關程式碼。交易產生器將測試運算分別兩種，第一種是關於操作性的問題，包括載入資料庫、備份資料庫、建立索引等。第二種使用者的交易運算，包括根據動態物件模型產生績效評估之測試運算。動態物件模型分成單一交易模型與複合交易模型。

1.操作性的問題

在定義資料庫綱目與經由資料庫產生器產生資料庫後，第一個要測量地便是載入資料檔案，再將其轉換成資料庫所需要的時間。載入的時間必須包含建立目錄（ catalogs ）、建立索引、備份新的資料庫到磁帶或其它設備上、及復原資料庫的內容等。

2.使用者的交易運算

使用者的交易運算依工作量模型中的交易模型種類分成兩種，分別爲單一的交易運算與複合的交易運算。單一交易運算是依據每種物件種類之狀態遞移圖依序作各個狀態遞移模擬，以測量其實際之效能。複合交易運算是依據不同物件之合作行爲作測試，此合作行爲乃是透過每個物件間的訊息傳遞與函式呼叫來達成目的。交易產生器必須依據文字型態交易模型之內容，透過對於語法與語意的分析，產生可以執行的程式碼。

㈢測試驅動器

測試驅動器是績效評估工作量模型的核心，負責統籌工作量模型之運作執行。其主要功能有 *1.* 根據控制模型的描述，順序來執行各筆交易。*2.* 在執行每筆交易間隔，負責清除快取記憶體的內容，以免因爲快取記憶而使測試結果受到影響。*3.* 根據使用者想衡量的資料庫績效係數爲基準，衡量物件導向資料庫的績效。

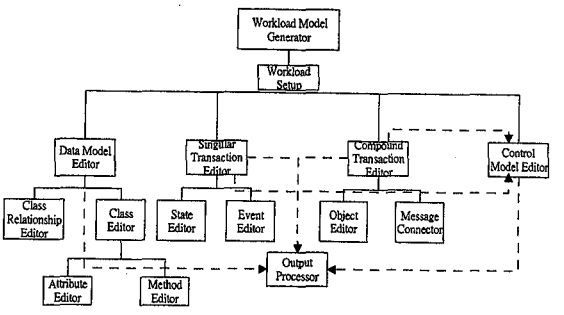

圖11：工作量模型產生器的程式結構

伍、系統實作

一、實作系統架構

本實作系統的架構，主要是讓使用者透過使用者介面建立自己的圖形型態工作量模型，包括資料模型、交易模型與控制模型。程式利用自己的資料結構與變數來儲存整個工作量模型，當使用者選擇系統輸出選項時，系統會將以變數儲存的工作量模型轉換成文字型態規格描述語言，實作系統的整體架構如圖10所示。由於實作系統主要能夠讓使用者編輯自己的工作量模型，並將圖形型態的工作量模型轉換成文字型態的規格描述語言，因此我們將實作程式稱之爲「工作量模型產生器」。

二、工作量模型產生器

由以上對於工作量模型產生器環境之描述，我們將工作量模型產生器分成五個子系統，分別爲「資料模型編輯器」、「單一交易模型編輯器」、「複合交易模型編輯器」、「控制模型編輯器」及「輸出處理器」，每個子系統又分別擁有各自的子系統，整個系統的分解圖呈樹狀的結構，如圖11所示。圖中的虛線代表資料的傳遞，控制模型編輯器因爲必須參考交易模型編輯器的交易資料。因此兩個模組間會有訊息的交換，四個子系統的模型資訊都必須傳送到輸出處理器上，以產生文字型態的工作量規格描述。

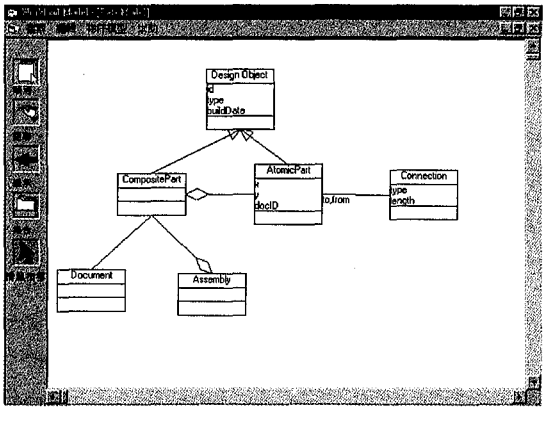

圖12：資料模型編輯器的的畫面

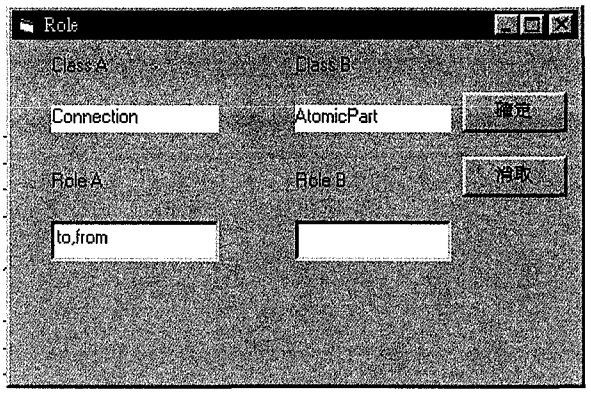

圖13：指定類別關連之角色的畫面

(一)資料模型編輯器

資料模型編輯器主要包括「類別關係編輯器」與「類別編輯器」。類別關係編輯器主要是爲了連繫類別間邏輯上的關係，包括繼承、集合與關連，透過繼承關係的使用，我們可以編輯出類別階層圖；使用集合關係，我們可以顯示類別與類別之間的集合關係；關連則可使類別之屬性值指向彼此的物件。圖12是利用資料模型編輯器所建構出來的部份 007 績效評估類別關係圖之畫面；此外，我們亦可對於類別間的關係作細部設計，如圖13是指定兩類別在關連中所扮演之角色的畫面。

類別編輯器則是用以編輯類別的細部格式，包括屬性編輯器與方法編輯器，使用者只要雙擊某個類別，即可進入類別編輯器的編輯畫面（如圖 14 所示），此畫面顯示類別的規格，包括類別的名稱、主要的鍵值及類別的所產生物件個數（資料筆數）等。另外使用者可以按下「增加」、「刪除」與「修改」等按鍵來爲此類別增加、刪除與修改類別的屬性與運算，屬性可以指明其名稱及型態，運算亦可指定其係數個數與型態、傳回值型態與儲存之檔案名稱與位置。當使用者於選項中指定所使用的編輯器後，使用者祇需按下「編輯」的按鈕，即可立即編輯每個物件的運算。爲了儲存以上每種類別的資訊，若以 VB 的語法來表示每一類別的資料結構，則如以下所示：

```
Pulic clss_name AS String
Public X As Integer
Public Y As Integer
Public width As Integer
Public height As Integer
Public attribute_num As Integer
Public operation_num As Integer
Public num_of_rows As Integer
Public primary_key As String
Private member_attribute ( 50 ) As attribute_entry
Private member_operation ( 50 ) As operation_entry
```

其中 class_name 儲存類別的名稱，X、Y 則用以表示其坐標，width、height 表示其寬度與高度，attribute_num、operation_num 則用以儲存其屬性與運算的個數，num_of_rows 記錄此類別的物件個數，primary_key 則用以記錄此類別的主鍵，member_attribute 的型態爲 attribute_entry，用以記錄類別中所包含

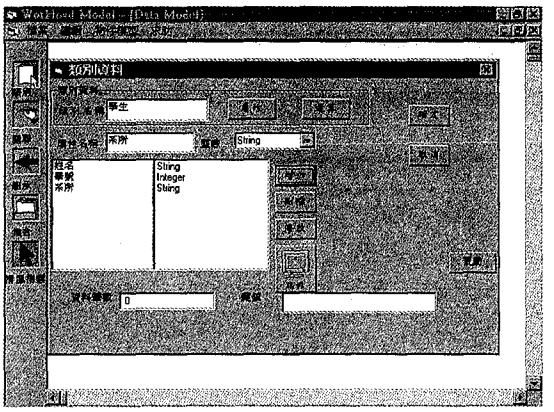

圖14：類別編輯器的畫面

的屬性之設計規格，member_operation 的型態爲 operation_entry ，用以記錄類別提供的運作之設計規格。attirbute_entry 與 operation_entry 的資料結構宣告方式如下所示：

```
Public Type attribute_entry
name As String
data_type As String
End Type
Public Type operation_entry
filepath As String
return_string As String
operation_string As String
End Type
```

(二)單一交易編輯器

單一交易編輯器主要是對於資料模型內的各個類別指定物件狀態的生命週期，包括編輯物件在其生命週期內不同的狀態與因爲事件之發生所產生的遞移情況。單一交易編輯器內又包含「狀態編輯器」與「事件編輯器」，如圖15是使用單一交易編輯器的編輯畫面。

狀態編輯器除了顯示類別的狀態遞移圖，且可讓使用者指定狀態的名稱，並增加或者刪除在此狀態內於不同時間下的行動，如圖 15 是狀態編輯器的編輯畫面。對於每個狀態我們使用以下的資料結構來予以儲存：

```
Public state_name As String
Public state_type As Integer
Public X As Integer
Public Y As Integer
Public width As Integer
Public height As Integer
Public action_num As Integer
Private action ( 50 ) As String

Public Sub set_action ( no As Integer , act
As String )
action ( no ) = act
End Sub

Public Function get_action ( no As Integer )
As String
get_action = action ( no )
End Function
```

其中 state_name 用以儲存狀態的名稱； state_type 用來記錄狀態的種類，我們在程式中將狀態的種類分成三種，分別爲 START 、 END 與 STATE ，分別表示啓始狀態、結束狀態與一般的狀態， x 、 y 則是用以儲存其畫面的坐標， width 和 height 則用以記錄其寬度與高度，另外亦分別以 action_num 與陣列 action 來儲存其狀態下的所有行動。由於在 VB 中陣列資料祇能以私有 (private) 的方式來宣告，因此，我們尚必須設定其存取的界面 set_action 與 get_action 。

事件編輯器可以讓使用者輸入事件的名稱、事件的係數、條件、行動、所引發的事件與係數與影響的物件等，事件編輯器之畫面如圖16，圖17所示。我們使用以下的資料結構來儲存事件之發生所引起的狀態遞移：

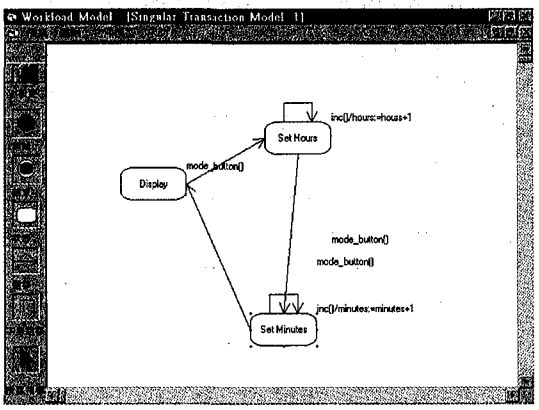

圖15：單一交易編輯器的編輯畫面

```
Public source As Integer
Public destination As Integer
Public relation_type As Integer
Public source_x As Integer
Public source_y As Integer
Public destination_x As Integer
Public destination_y As Integer
Public serial_num As Integer
Public roleA As String
Public roleB As String
Public message_string As String
Public event_entry As eventClass
```

狀態遞移所宣告之資料結構亦使用在記錄上述類別間的關係與物件合作圖之訊息傳遞，因此有些資料結構是爲了儲存類別間關係與物件訊息的特徵，例如 relation_type、roleA 與 roleB 是爲了記錄類別間的關係。source 與 destination 是爲了記錄狀態遞移之來源與目的狀態。source_x、source_y、destination_x、destination_y 則用以記錄來源狀態與目的狀態的連線坐標，主要爲了畫面的美觀目的；serial 則是用以記錄物件訊息傳遞時的序號；roleA 與 roleB 用以記錄類別關係時，兩個類別所扮演的角色。message_string 則是記錄訊息的字串，它是由 event_entry 結合所有事件的設定所得到的字串，eventClass 的資料結構如下所示：

```
Public event_name As String
Public augument As String
Public condition As String
Public action As String
Public object As String
Public send_event_name As String
Public send_event_augument As String
```

上述的資料結構分別用以儲存事件的名稱、事件的係數、條件、條件成立後的

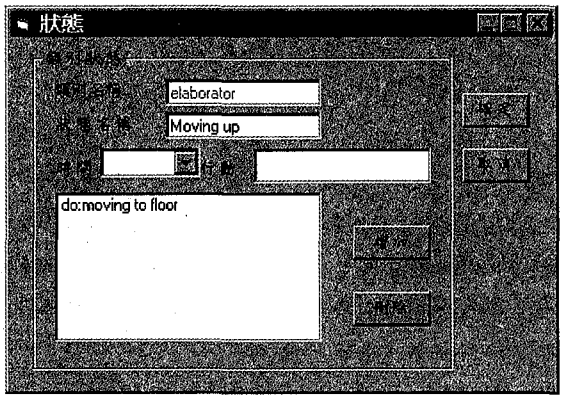

圖16：狀態編輯器的編輯畫面

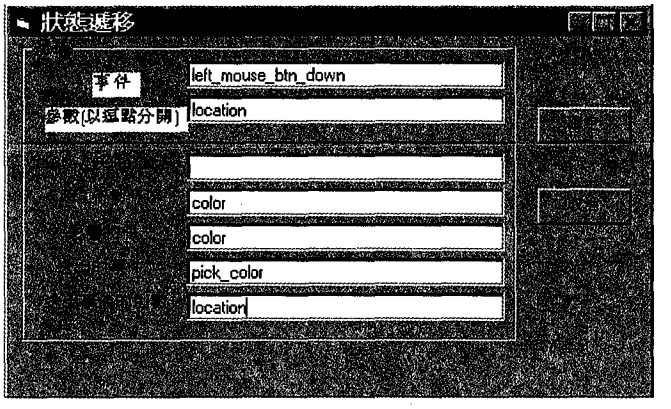

圖17：事件編輯器的編輯畫面

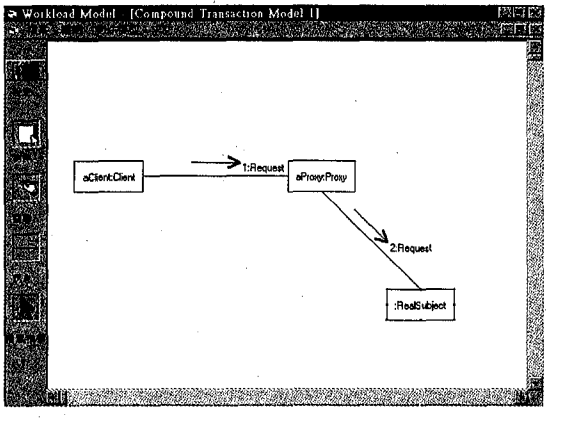

圖18：代理器的物件合作圖編輯畫面

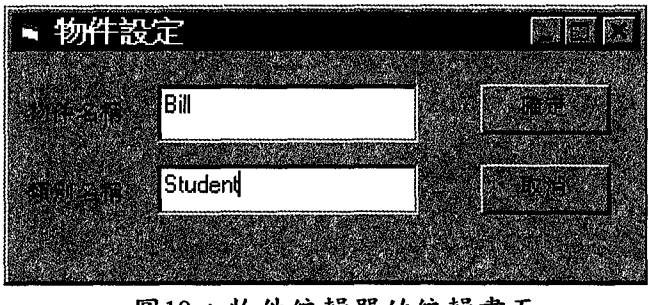

圖19：物件編輯器的編輯畫面

行動、所影響的物件、狀態遷移所引發的事件名稱及其係數。

㈢複合交易編輯器

複合交易編輯器讓使用者編輯不同物件間的訊息傳遞關係，以達成邏輯上交易目的，複合交易編輯器包含「物件編輯器」與「訊息編輯器」。物件使用 object_name 、 class_name 來記錄，訊息則與類別關係與事件共享一個資料結構。

物件編輯器可以讓使用者可以指定物件的名稱與屬於的類別，其編輯畫面如圖19；訊息編輯器可以讓使用者指定訊息的序號、名稱、係數與同步性的種類等，編輯畫面如圖20所示。

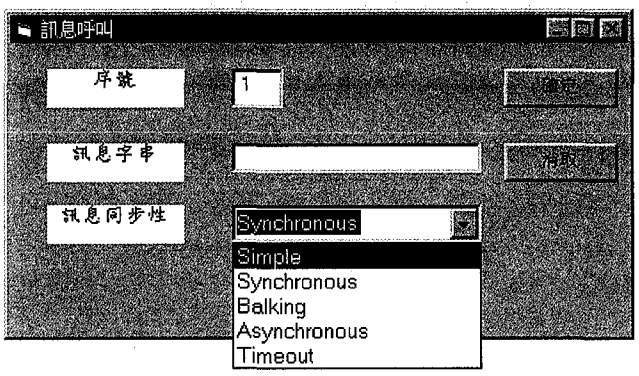

圖20：訊息編輯器的編輯畫面

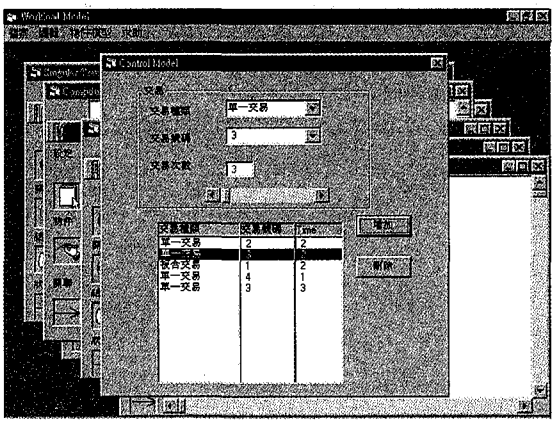

圖21：控制模型的編輯

㈣控制模型編輯器

控制模型編輯器主要能夠讓使用者對於已輸入的交易，指定各交易在績效評估測試時，其交易的順序與數目。控制模型編輯器的主要畫面如圖21所示。

㈤輸出處理器

當使用者編輯完成工作量模型或祇完成部份工作量模型時，使用者可在系統主選單中選擇「檔案輸出」選項，輸出處理器即會將圖形型態工作量模型轉換成文字型態工作量模型，以利使用者下一步驟的應用。系統的輸出畫面如圖22所示。

陸、結論與未來的研究方向

在本文中，我們提出了以物件導向系統分析方法來建構開放型使用者績效評估之工作量模型，解決一般績效評估方法因只針對特定應用領域，無法周詳考量到使用者的特定需求問題。使得測試工作量對於使用者而言更具一致性、代表性與正確性，測試之結果對於使用者而言更具意義。同時舉出如何使用文字型態的語言描述來對映圖形型態的工作量模型，以利電腦內部的儲存與處理和增加工作量模型之可攜性。我們亦說明了如何將文字型態的工作量模型對映至需績效評估應用程式，及績效評估實驗設計和執行的三個模組元

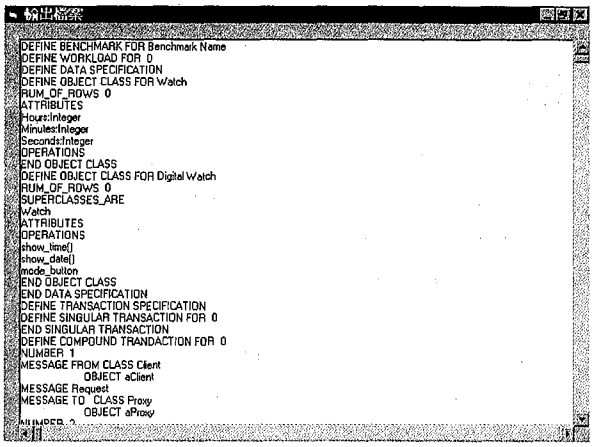

圖22：工作量模型產生器的輸出畫面

件的概念性架構。

以使用者需求爲導向的績效評估方法將會是未來物件導向資料庫之績效評估的發展趨勢，可以從以下幾個觀點得知：

1. 在物件導向資料庫技術愈趨於成熟時，使用物件導向資料庫的企業或個人將會愈來愈多，應用的範圍也愈來愈廣。許多物件導向資料庫績效評估的假設，諸如使用者將物件導向資料庫應用於工程領域等假設將不再合時宜。唯有根本從使用者之應用需求面著手，以建構使用者本身的工作量模型，才能使測試結果之數據對於使用者而言有眞正的意義。
2. 在物件導向之系統分析方法逐步歸於一致，不同方法之異質性逐漸消除時，使用者將更容易使用物件模型裡的圖形符號來建構自己的工作量模型。
3. 資料庫的績效評估其資料庫綱目演變得愈來愈複雜、測試運算趨於繁多，將使得使用者欲根據規格自行實作資料庫績效評估的過程佈滿荆棘。一套有系統、易於實作的方法加上工具的支援將會大大地減少績效評估之人力、物力成本，亦將會未來資料庫績效評估的發展方向。

在本論文這一階段的研究發展過程中，我們發現在完成本階段的研究工作之後，關於物件導向資料庫之工作量模型建構的研究工作還可以朝下列的方向繼續進行：

1. 對於分散式物件導向資料庫作績效評估的研究。分散式物件導向資料庫將會是未來資料庫的趨勢之一，原因在於分散式的處理，不但可以分散與加快資料處理，且分散式的資料庫正反應出企業本身邏輯上與實質上的分散性，區域性的資料存放在邏輯上最相關的地方，而在需要時也能取用遠地的資料。
2. 建構多個使用者的工作量，並作實際的測試工作。資料庫系統的使用往往是同時有多個使用者在資料庫上進行交易。但現今的資料庫績效評估的研究往往祇偏向單一使用者的工作量作測試，主要的原因在於多個使用者之工作量構面更加廣泛，其工作量又牽涉到多個等時活動的複雜交互作用。
3. 對於文字型態物件模型轉換成績效評估之應用程式之過程，可以作進一步的研究。本研究之實作部份主要是將使用者之圖形型態工作量模型轉換成文字型態的工作量模型，對於將文字型態工作量模型轉換成實際可執行的績效評估應用

程式則未加以驗證。

參考文獻

1. Anderson, T. et al., "The HyperModel Benchmark", Lecture Notes in Computer Science: Advances in Database Technology Proceedings, EDBT'90, 1990.
2. Bailin, S. C., "An Object-Oriented Requirements Specification Method", Communcations of the ACM, May 1988, pp.608-623.
3. Beck, K., "A Laboratory for Teaching Object-Oriented Thinking", OOPSLA'89 Proceedings, October 1989, pp.1-6.
4. Bitton, D. and C. Turbyfill, "A Retrospective on the Wisconsin Benchmark", Readings in Database Systems, Ed. By M. Stonebraker, Morgan Kaufmann, Inc., 1998.
5. Booch, G., Object Solutions: Managing the Object-Oriented Project, Benjamin /Cummings Publishing Company, Inc., 1996.
6. Booch, G., Object-Oriented Analysis and Design with Applications, Benjamin / Cummings Publishing Company, Inc., 1994.
7. Booch, G., "Object-Oriented Development", IEEE Transaction on Software Engineering, February 1986, pp.211-221.
8. Carey, Michael J., David J. Dewitt, Chander Kant, and Jeffrey F. Naughton, "A Status Report on the OO7 OODBMS Benchmarking Effort," In Proceedings of the ACM OOPSLA Conference, Portland, OR, Octorber 1994, pp.414-426.
9. Carey, Michael J., David J. Dewitt, Jeffrey F. Naughton, Mohammad Asgarian, Paul Brown, Johannes E. Gehrke, and Dhaval N. Shah, "The BUCKY Object-Relational Benchmark", In Proceedings of 1997 ACM-SIGMOD International Management of Data, May 1997.
10. Carey, Michael J., David J. DeWitt,and Jeffery F. Naughton, "The OO7 Benchmark", In Proceedings of the 1993 ACM-SIGMOD Conference on the Management of Data, Washington D.C., May 1993, pp.12-21.
11. Cattell, R. Cattel and J. Skeen. "Object Operations Benchmark," ACM Transactions on Database Systems, Vol. 17, No. 1, March 1992, pp.1-31.
12. Cattell, R. G. G., The Object Database Standard: ODMG-93, Release 1.1. Morgan Kaufmann, San Francisco, 1994.
13. Cattell, R. G. G., The Object Database Standard: ODMG-99, Morgan Kaufmann, San Francisco, 1999
14. Ceri, Stefano and Piero Fraternali, Designing Database Applications with Objects and Rules, Addison Wesley, 1997.
15. Duhl, J. and C. Damon, "A Performance Comparison of Object and Relational Databases Using the Sun Benchmark", OOPSLA Proceedings, 1988, pp.153-161.
16. Gertino, E. and L. Martino, Object-Oriented Database Systems, Addison Wesley, 1993.
17. Gray, Jim, The Benchmark Handbook, Morgan Kaufmann, San Mateo, CA,

1993.

18. Jackcoson, Use Case Object-Oriented Software Engineering, Addison Wesley, 1997.
19. Kerth, N. L., "A Structured Approach to Object-Oriented Design", OOPSLA'91, Addendum to the Proceedings, pp.21-42.
20. Kilov, H., "Reviews of Object-Oriented Papers", Part I, ACM SIGMOD Record, Vol. 18, No. 1, 1989, pp.12-15.
21. Kilov, H., "Reviews of Object-Oriented Papers", Part II, ACM SIGMOD Record, Vol. 18, No. 4, 1989, pp.12-15.
22. Kim, W., "Object-Oriented Databases: Definition and Research Directions", IEEE Transaction on Knowledge and Data Engineering, Vol. 2,No. 3, 1990, pp. 327-341.
23. Kim, W., "Observations on the ODMG-93 Proposal for an Object-Oriented Database Language", ACM SIGMOD Record, Vol. 23, No. 1, 1994, pp.4-9.
24. Lee, S. and D.L Carver, "Object-Oriented Analysis and Specification: a Knowledge Base Approach", Journal of Object-Oriented Programming, January 1991, pp.35-43.
25. Martin, J. and J. Odell, Object-Oriented Analysis and Design, Prentice-Hall, 1992.
26. Meyer, B., "Reusability: The Case for Object-Oriented Design", IEEE Software , March 1987, pp.50-64.
27. O'Neil, Patrick E., "Revisiting DBMS Benchmarks", Datamation, Sept. 15, 1989 , pp.47-54.
28. Rao, Bindu Rama, "Distributed Applications? Don't Forget the Database", Data Communications, October 1995.
29. Rational Software Corporation, Unified Modeling Language, 1997-1999.
30. Rubenstein, W., M. Kubicar, and R. Catell, "Benchmarking Simple Database Operations", In Proceedings ACM SIGMOD Conference, San Francisco, California, May 1987.
31. Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, Object-Oriented Modeling and Design, Prentice-Hall, Inc., 1991.
32. TPC-C, Transaction Processing Performance Council, TPC Benchmark C Standard Specification, August 1995.
33. TPC-H, Transaction Processing Performance Council, TPC Benchmark H Standard Specification, August 1998.
34. TPC-R, Transaction Processing Performance Council, TPC Benchmark R Standard Specification, August 1998.
35. TPC-W, Transaction Processing Performance Council, TPC Benchmark W Standard Specification, August 1999.
36. Thatte, J. S., C. Thompson and D. Wells, "Report on the Object-Oriented Database Workshop", ACM SIGMOD, Vol.18, No. 3, 1989, pp.78-101.
37. Vossen, G., "Bibliography on Object-Oriented Database Management", ACM SIGMOD Record, Vol. 29, No. 1, 1991, pp.24-46.
38. Wilkie, George, Object-Oriented Software Engineering, Addison-Wesley, 1994.
39. ao, B. S., "Requirements Analysis and Database Benchmark," Guest Speech Notes, University of Maryland, College Park, Maryland USA, 1992.
40. Yao, Hevner, Meyer, "Benchmarking

Distributed Database Machines," IEEE TOSE, 1987.

41. Yao and Hevner, "Query Optimization of Distributed Databases," IEEE TOSE , 1982.

42. Yu, "OBJECTIVE Active Database Benchmark," Journal of Systems and Software, 1999.

43. Yu, Philip S., Ming-Syan Chen, Hans-Ulrich Heiss, and Sukho Lee, "On Workload Characterization of Relational Database Environments", IEEE Transactions on Software Engineering, Vol 18, No.4, April 1992, pp.347-355.