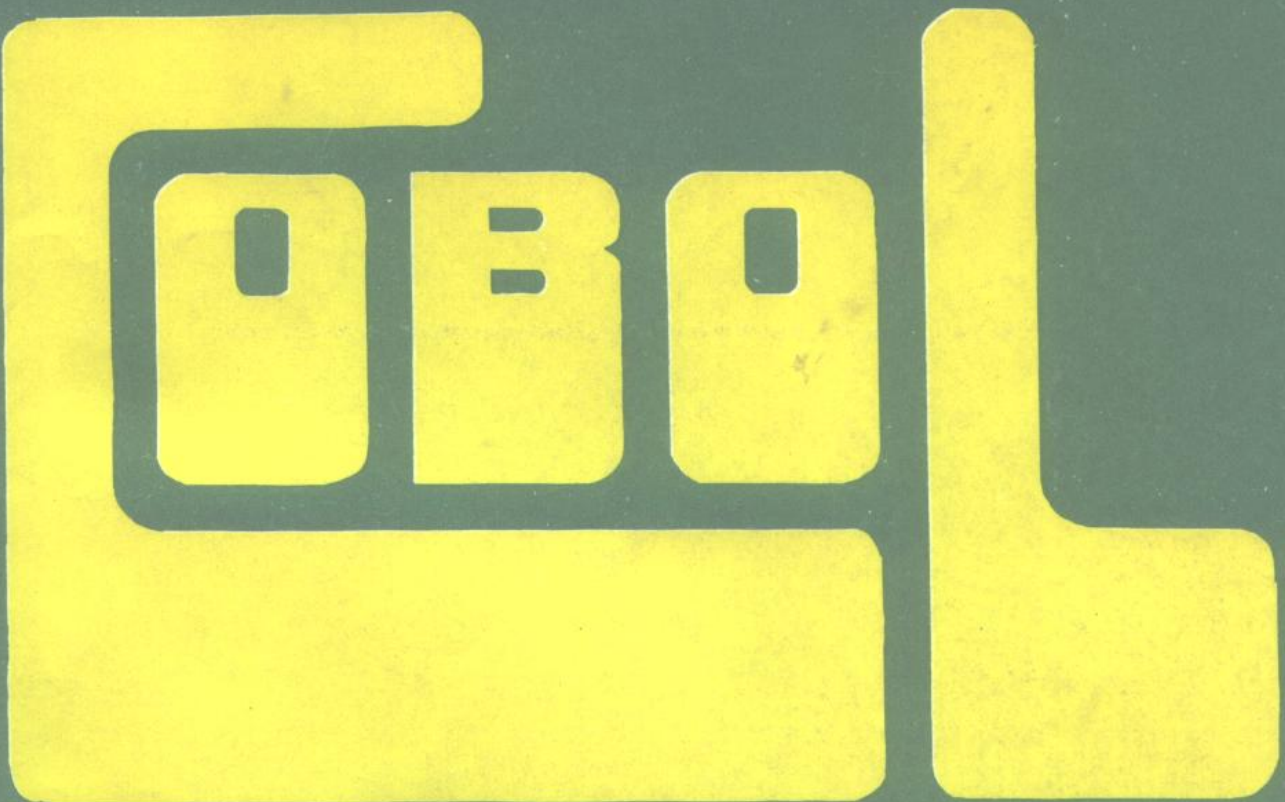


COBOL

档案技术与资料库

叶耀明 编著

A large, bold, yellow graphic of the word "COBOL" in a stylized, blocky font. The letters are interconnected, with the 'O's and 'B' sharing vertical strokes. The graphic is set against a dark green background.

COBOL

A small, partially visible red logo in the bottom left corner, featuring a stylized white character.

松岗电脑图书资料有限公司

75-100
7-11

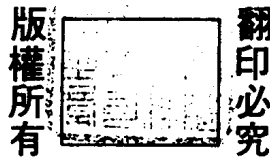
COBOL

檔案技術與資料庫

葉耀明 編著

松崗電腦圖書資料有限公司 印行

COBOL
檔案技術與資料庫



每本定價 220 元整

書號：210132

編著者：葉 耀 明

發行人：吳 守 信

發行所：道明出版社

台北市仁愛路二段一一〇號三樓

總經銷：松崗電腦圖書資料有限公司

台北市仁愛路二段一一〇號三樓

電話：3930255 · 3930249

郵政劃撥：109030

印刷者：泉崗印刷設計股份有限公司

台北市仁愛路二段一一〇號三樓

電話：3930255 · 3930249

中華民國七十年七月 第一版

中華民國七十二年二月 第二版

本出版社經行政院新聞局核准登記，

登記證號為局版台業字第一七二九號

序

也許有些讀者於學了一學期的COBOL電腦語言後，還不曉得COBOL語言在商業方面被廣泛採用的主要原因。總認為COBOL語言在處理數學運算及應用到一般的演算法(Algorithm)時，不如FORTRAN, PASCAL簡捷方便。殊不知COBOL語言的最大特點乃在資料與檔案處理方面，檔案處理可謂COBOL的精粹靈魂所在。是故，學COBOL的人，不可不知檔案技術。

當讀者學得基本COBOL語言後，雖已能寫寫簡單的習題，然一旦面臨檔案處理就束手無策，因學校從來沒開過這門課，且坊間也找不到專論此類的書籍。只能經由老前輩的口中，一點一滴地套問出他們以血汗所掙來的經驗談。而這些經驗談又不乏以偏概全，指鹿為馬的說法。因他們的經驗大都來自親自參與企業的檔案設計作業，有很多觀念為專屬於此企業應用的特例，無法適合一般的檔案系統，有時他們本身甚至誤解了檔案系統的設計概念及其重要決定因素。基此，市面極需要一本專論檔案技術的書，澄清人們的觀念，闡明如何做好檔案處理與設計工作。本書乃有鑑於現今學生與專業人員的此項需要而編撰的。

本書可作為一般大專程度的COBOL程式語言第二學期的檔案處理課程的教學或參考之用；也提供專業人員學習檔案技術獨立閱讀之用。本書內容，於檔案技術方面得自“File Technique for Data Base Organization in COBOL” Leroy F. Johnson and Rodney H. Cooper(1981)。

章節安排也依此書為準。此書作者對檔案技術有獨到精闢的了解，提出的論點乃1981年世界最新發展趨勢。

本書提出COBOL電腦語言在檔案處理與資料庫管理方面的使用技巧，及如何

將資料結構應用到檔案技術，然後闡明如何適當設計檔案系統。綜觀全書，以檔案為經，COBOL 為緯，經緯交錯連貫，相輔相成。書中檔案技術的討論，以企業之個案研究為基準，乃強調檔案技術理論與實際應用相互配合，望讀者莫忽視這些個案研究。

謹以本書獻給嚴父慈母，葉坤倫先生，葉黃素月女士。感謝父母親多年寒暑教導養育之情，雖無法寫出慈母頌或背影等等動人心腑，真情流露之文字，祇以本書之出版感激親思之萬一。

葉耀明

七十年八月

簡介

由於電腦科技的充分發達，其領域已趨向於資訊的處理。爲了有效又經濟地利用資訊，而發展爲將資料與使用此資料的程式互相獨立。如此一來，使得資料的價值隨之提高，因爲不同的程式可以共用相同的資料。爲了達到此目的，發展出資料管理技術（DATA MANAGEMENT），使程式設計師（programmer）於發展設計程式之時不需牽涉到資料的實體結構與資料存於資料庫的邏輯構造。

雖然，資料庫的使用，可減輕程式設計師的工作，使其不須建實體檔（physical file），但他還是必須建立爲他所用的邏輯檔（logical file）。由於目前資料庫處理資料還無法如理想狀況那樣爲所欲爲，因此對於實體檔結構的了解實爲程式設計所需要的知識。同樣地，對檔案技術的了解也相當重要，因爲程式的應用往往無法完全的一般化且無法完全與實體結構相獨立。換句話說，程式設計師到最後還是必須建實體檔來存於資料庫之內。因此目前大型資料庫系統的發展，一致強調檔案技術（File technique）的重要性。此方面亦即本書所詳細討論的重心所在。

實際而言，要做與語言獨立的運算（language-independent computing）並非不可能。但在尋求一般化的過程之中，最好能夠對特殊的需求有深刻的了解。有一個門徑，即是利用“虛碼”（pseudo code）。對檔案技術的學習而言，虛碼的使用並不超越一般程式語言（programming language）的應用，二者都同樣的重

要，在本書中我們利用 COBOL 語言來介紹檔案技術的使用，這並非我們對 COBOL 有偏愛，而是因 COBOL 為商用語言，在此方面有較明確有效的處理，且“設計”(design)——即是對於實際世界的種種限制之中尋求出一個最好的決擇以資利用。讀者可以將 COBOL 視為 於商業社會之中所碰到的一個實際系統，由此系統而尋找出檔案處理技術的最好方法。在實際的檔案處理技術之中，其困難之處並非在於技術本身，而在於其所需要資料分佈於不同的來源。因此本書的目的即將資料集中到一個來源，以適當的 COBOL 指令做定義與使用檔案，而介紹邏輯檔與實體檔的各種類型的應用。本書中提供了各樣式的例子，而不單純的指出法則，以告訴讀者為何且如何做好檔案的設計。此乃因為，吾人的方向在於提供讀者穩固的基礎背景與足夠的細節了解，使在面對 COBOL 環境時，可以立即工作；而不是要讓讀者於唸了本書之後馬上成為專家，但至少可使諸君於碰到檔案問題時，於合理的時間範圍之內得以完善解決。一方面，本書可視為 COBOL 教科書的媒介；另一方面，則為介紹如何利用一個語言的基本指令來擴充到檔案處理技術。因此諸君可以於接觸了一學期的 COBOL 程式技巧之後，為求利用 COBOL 有效地處理檔案，而參考本書，相信其所涵蓋的內容可讓諸君學有所得，裨使各位開卷有益。

COBOL

本書所用 COBOL 語言定義於 ANSI X3.23-1974。此型種比一般市面所見的書所使用 ANSI COBOL 型種為新。

由於 COBOL 程式並不能脫離電腦而獨立存在，因此吾人在本書中例子所使用編譯器(compiler)為 IBM OS/VS COBOL compiler，所用的系統控制語言(system control language)為 IBM 370 系統的 JCL，如此選擇有兩種理由：其一為，此系統乃為吾人執行本書各例子中 COBOL 程式的系統；其二為，此系統曾被社會各界所使用。在本書中，於必要時會特別指出標準 ANSI COBOL 與我們例子中所用 IBM 系統之 COBOL 的差異，方便於讀者做比較。

目錄

簡介

第一章 概論	1
1-1 資料	3
1-2 檔 (File)	7
1-3 邏輯組織	12
1-3-1 線性結構	12
1-3-2 非線性結構	17
1-3-3 語言資料結構	19
1-4 實體組織	20
1-5 資料作業	22
1-6 資料庫	25
1-7 資料檔的沿革與展望	27
第二章 資料管理	29
2-1 記憶裝置	30
2-1-1 磁帶機	33
2-1-2 磁帶盤	36
2-2 集區 (blocking)	36

2-3	緩衝 (buffering)	38
2-4	檔的使用	41

第三章 COBOL 循序檔 41

3-1	緒言	41
3-2	個案研究：音樂日誌	43
3-2-1	邏輯分析	43
3-3	COBOL 循序輸入 - 輸出模式	46
3-4	循序檔作業	51
3-5	平衡線法 (Balance Line Algorithm)	54
3-6	個案研究：會計簿記	56
3-6-1	邏輯分析	57
3-6-2	PROFIT 公司會計簿記系統	60
3-7	個案研究：飛彈目標系統	73
3-7-1	邏輯分析	73
3-8	變長錄	79
3-8-1	COBOL 變長錄	79
3-8-2	飛彈目標系統之變長錄	81

第四章 分類 89

4-1	緒言	89
4-2	內部分類	90
4-2-1	插入法 (Insertion Sort)	91
4-2-2	交換法 (Interchange Sort)	92
4-2-3	快速法 (Quick Sort)	93
4-3	外部分類	95
4-3-1	磁盤分類	95

4-3-2	磁帶分類	97
4-4	COBOL 提供的分類	99
4-4-1	基本分類	99
4-4-2	特定錄分類	103
4-5	合併 (Merge)	107
4-6	結論	111

第五章 COBOL 相對檔 113

5-1	緒言	113
5-2	個案研究	115
5-2-1	邏輯分析	115
5-3	相對檔處理：車輛牌照檔	116
5-4	相對檔循序出入	123
5-5	相對檔雜亂出入	133
5-6	結論	139

第六章 錄位址技術 141

6-1	緒言	141
6-2	位址決定	142
6-3	搜尋 (Search)	144
6-4	更新目錄	147
6-5	位址計算	149
6-5-1	鍵 - 即 - 位址法	149
6-5-2	鍵轉換演算法	149
6-5-3	雜湊法 (Hashing)	150
6-5-4	除法雜湊 (Division Hashing)	150
6-5-5	雜湊碰撞	152
6-5-6	解決雜湊碰撞	153

6-5-7 開於位址法 (Open Addressing)	154
6-5-8 鏈結法 (Chaining)	163
6-6 其他位址技術	165
6-7 結論	166
第七章 IBM 直接檔	167
7-1 緒言	167
7-2 循序建直接檔	168
7-3 雜亂出入直接檔	176
7-4 結論	186
第八章 COBOL 指標檔	187
8-1 緒言	187
8-2 個案研究：信託合作社	189
8-2-1 邏輯分析	189
8-2-2 指標檔作業	190
8-3 指標輸入 - 輸出模式	193
8-3-1 設備部 (Environment Division)	194
8-3-2 資料部 (Data Division)	195
8-3-3 處理程序部 (Procedure Division)	196
8-3-4 指令總整理	206
第九章 COBOL 指標實體組織	209
9-1 緒言	209
9-2 指標循序組織	210
9-3 ISAM之資料實體組織	213
9-4 ISAM溢出錄處理	217

9-5	建 ISAM 檔	222
9-6	VSAM	222
9-6-1	錄組織	224
9-6-2	控制段 (Control Interval)	225
9-6-3	建 VSAM 檔資料集	228
9-7	結論	230

第十章 資料結構 231

10-1	定態資料結構	232
10-2	動態資料結構	234
10-2-1	線性結構	234
10-2-2	非線性結構	237
10-3	鏈結串列	240
10-3-1	線性鏈結串列	240
10-3-2	非線性鏈結串列	246
10-4	檔案結構	247

第十一章 檔案查詢 249

11-1	個案研究：汽車出租系統	249
11-1-1	邏輯分析	249
11-2	複合查詢	253
11-3	TRACE 演算法	256
11-4	錄組織技術	259
11-5	二元列表 (Binary Table)	260
11-6	結論	264

第十二章 安全性與完整性 **265**

12-1 資料完整性 (Integrity)	265
12-1-1 外部確認 (External Validation)	266
12-1-2 內部確認 (Internal Validation)	267
12-2 資料安全性 (Security)	270
12-2-1 內部安全 (Internal Security)	270
12-2-2 外部安全 (External Security)	273
12-3 結論	274

第十三章 檔案設計 **275**

13-1 檔案組織	275
13-2 實際考慮	282
13-3 檔案設計標準	285
13-4 檔案設計步驟	286
13-5 個案研究：失竊汽車檔	288
13-5-1 邏輯分析	288

第十四章 系統設計與完成 **299**

14-1 系統設計通論	299
14-1-1 設計品質	303
14-1-2 文件記錄	303
14-2 程式師 (Programmer)	304
14-3 層次規劃 (Structured Programming)	306
14-4 測試與程式確認	311
14-5 程式及設計之其他細節	312
14-6 結論	314

第十五章 資料庫系統

315

15-1 準技術考慮	315
15-1-1 資料定義語言——DDL	317
15-1-2 資料處理語言——DML	317
15-1-3 資料庫管理者——DBA	318
15-1-4 資料庫管理系統的結構	318
15-2 資料庫系統的要求與目的	319
15-3 資料模式(Data Model)	322
15-4 CODASYL 資料庫模式	326
15-5 CODASYL 資料描述語言(DDL)	327
15-6 CODASYL COBOL 資料庫設備	329
15-6-1 資料庫錄及其相關術語	330
15-6-2 錄選定	330
15-6-3 DML 指令	331
15-7 結論	333

附錄

附錄 A	335
附錄 B	337

第一章 概論

在開始討論本書的內容之前必須對本書涵蓋的最基本元素——資料 (data)，做一個明確的定義與說明。何謂資料？其與運算的關係為何？雖然此為相當基本的問題，但要回答此問題時既不簡單，也無固定的答案。對於本書所討論的方向而言，我們可定義：

資料 (data)：為存在程式外面，被程式處理的一種情報 (Information)。由於主記憶體內方便於執行運算，而大部份時間都存於外界，副記憶體 (secondary memory) 之內，例如磁盤，磁帶之中。

起先，早期的程式設計，存於記憶體的資料只專屬於其特定的程式，不能被別的程式所共用。而目前的演變已慢慢傾向於資料與處理此資料的程式相互獨立，此演變可歸納為幾種因素：

- (一) 程式系統日趨複雜，相同的資料往往反覆出現於不同的檔 (file) 與不同程式之內。
- (二) 由於資料的大量需求，花費在收集、存檔與使用所需求資料的費用，已成為整個系統的主要開支。
- (三) 維護資料時無法避免的資料結構上之改變，希望能儘量不影響到其相關的程式系統，以減少軟體維護的費用。

2 COBOL 檔案技術與資料庫

以上觀念的進步，對資料存放技術之發展有顯著助益。由於資料維護成本不斷提高，吾人慢慢注意到資料的保護，使其免於結構上之經常更換。如此就必須發展資料存放一般化。基於此了解，已迫使吾人對資料有效、成本低的運算做重新考慮，進而面對到資料管理的問題。

由於技術不斷改進，以前認為有效的解決方法，目前已被淘汰。電腦程式，就如汽車一樣，會隨使用時間的流逝而退化，到最後，維護舊車的費用比買一輛新車還貴。因此在解決電腦所面臨的問題時，往往將舊程式丟棄，重新設計，反而可得較好的效果。不幸地，在資料維護上往往面臨無法負擔舊檔案系統的維護費，也無法承受更新系統的費用，造成進退兩難無所適從的狀況。為解決此困境，因而引進了資料庫的觀念。定義名詞如下：

資料庫 (Data Base)：資料檔的集合，檔與檔之間有關係存在，可同時為多個應用程式所使用。

資料庫管理系統 (data Base Management System, DBMS)：包括資料庫本身以及為維持資料檔之間相互關係所須的軟體程式。

這些定義稍嫌簡單，但相當清晰，足夠目前需要。

基於此傾向，本書的重心即在於檔案處理的原則與技術、如何以 COBOL 來解決檔案問題以及檔與資料庫二者相互的關係。

在設計檔案時，一個問題常被提出：

如何處理資料與程式之間的關係？

吾人可以用三點來回答：

- (一) 對於不同之程式，資料的使用，越簡單越好。
- (二) 資料跟程式相互獨立。
- (三) 所用的資料易於查尋。

並非我們所述的技術都能滿足此三種要求，但也儘量符合這些原則了。

1-1 資料

電腦時代的來臨，提供了處理大量資料的能力。收集大量資料是一回事，但是否能夠有意義、適當的處理資料又是另外一回事。在商業社會中，因為資料處理系統工作的不當而倒閉破產的大有人在。因此在收集大量資料之後，還必須付出可觀的心血於資料組織結構方面，才能夠有效的處理這些資料。電腦運算的目的，並非在於資料的收集，而是為了幫助吾人做決策。如果一個決策並非偶然靈感的觸發與不合邏輯之決定，則其必為基於各來源的資料情報而做的最好選擇。在有些狀況下，這些策略可經由電腦適當的演算法（Algorithm）分析出最好的決定。可舉一例說明：當碰到客戶的帳單過期 60 日還未繳費，即可由電腦自動決定，送出信件來追繳費用，或向客戶下警告，威脅要尋求法律途徑解決。此即為電腦幫人類所作最簡單決策。

成本與收益的取捨

在設計資料處理系統時，常事先估計可能得到的收益與必須花費的成本將二者做一比較取捨，以決定最適當的系統。一般而言，理論上最優良的系統，往往並非合乎使用、最適當的系統。因理論上最優良的系統為滿足其理想，常不可避免地提高了系統成本，雖有收益的提高也不一定能補償成本的差額。故必須借助成本與收益的取捨分析，方能決定較適合使用的系統。

資訊財產

在審核一公司的資產時，會計師只考慮每樣物品其可換算的金錢價值；然而以經理的眼光來看，則著重於其手下職員編制為其資產的標準。當然，對一個證券市場分析家而言，此兩個因素都為其評定公司資產的標準。但有很多公司其所擁有最珍貴資產並非表現在收支圖表與組織編制之上，而是其在本行業務上所擁有的知識與資料。