講　　座

データベースの実際（1）†

——造船業における生産管理用データベースの一例——

窪田八洲洋†† 田中裕子††
中村洋四郎††† 古河建純†††

1. はじめに

最近，コンピューター利用技術が発達し，高度で多様な業務が要求されるに従い，データベースの必要性が叫ばれてきた．このような状況に応じて，各メーカー，ソフトウェア会社はデータベースのための様々なファイルマネージメントシステムの開発を盛んに行ない，その思想，技法，メリットについて紹介している．しかし，それらを用いた業務，システム開発の具体的解説は，今まで数少い．本論文では，データベースを中心とした業務統合化，ジョブ集約化，データの集中管理など，従来いわれている，ホスト言語型ファイルマネージメントシステムのメリットについて，具体例による実績値により，評価を与え，データベースを用いたシステムの設計，適用によるメリットなどを，より具体的に説明することを目的としている．本論文では，以上の主旨に沿い，最初に富士通（株）において開発された汎用ファイルマネージメントシステム，RAPID†††† の概要を紹介し，次に，川崎重工業（株）船舶事業本部において，RAPID を用い，設計，開発されたデータベースシステムの概要と適用過程並びに使用実績について述べる．

2. RAPID

2.1 概要

RAPID は統合ファイル（データベース）を容易に設定し，使用者のアプリケーションプログラムとは独立にデータベースの取り扱いを可能とする手段を使用者に提供することにより，煩雑なデータ管理業務から使用者を解放することを目的に開発されたものである．RAPID の開発にあたって次のことを設計方針とした．

- ○ 特定の業種を限定せず，汎用的に使用できるシステムとすること．
- ○ データベースを扱う RAPID 言語はコンパイラーインターフェイス（COBOL, FORTRAN, ALGOL）とし，使い易さを目指すこと．
- ○ ダイレクトアクセス装置を高度に利用できるようにシステム設計すること．
- ○ データ構造は階層構造をベースとし，使用者の自由な構造が可能で，データベースの作成，更新，検索が容易であること．

以上の設計方針にそって開発された RAPID は，次のような適用メリットを挙げることができる．

- ○ 総合的，管理的情報要求を効率的に処理可能．
- ○ 集中的なファイル管理が可能．
- ○ データの重複を最小限におさえる事ができる．
- ○ プログラミングコストの低減が計れる．
- ○ オペレーティングコストの低減が計れる．

このように RAPID はデータの統合管理システム実現に有力な TOOL としてさまざまな機能を持ち，効果的に使用することができる．RAPID の具体的使用例としては，生産，工程管理，部品供給管理，証券情報管理，営業情報管理などさまざまな分野での適用を挙げることができる．

2.2 データベースの構造

（1） 論理構造

RAPID のデータベースは**リング構造**を採用するこ

---

† An Example of Production Control System based on Data Base, by Yasuhiro Kubota, Hiroko Tanaka (KAWASAKI Heavy Industries, LTD. Ship Group, Planning And Control Office), Youshirô Nakamura and Tatsuzumi Furukawa (FUJITSU, LTD. System Research and Development Division)

†† 川崎重工業（株）船舶事業本部企画室

††† 富士通（株）電子システム開発部　第2システム部

†††† FACOM 230-60 用の RAPID (Retrieval And Production for Integrated Data；昭和46年1月提供)．富士通では，この RAPID の経験を基に，CODASYL DBTG (Data Base Task Group) の思想を反映して，FACOM 230-45S/55 用の RAPID（昭和47年11月提供）を開発した．また時系列データの扱いなどを容易に可能とした FACOM 230-60/75 用 FMS (File Management System；昭和47年11月提供) を開発している．

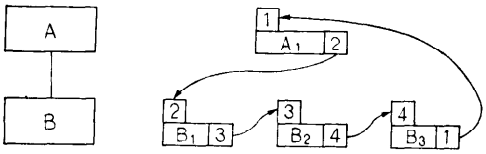


図 1　データの階層関係とリング

とにより，簡単な Tree 構造から Network 構造まで，複雑な論理構造を取扱うことが可能である．データベースの論理上の構成体はデータ種別により定義されたレコードタイプであり，レコードタイプ間に階層性を与えるために RAPID はリングを用いる．図1ではレコードタイプAとBとの階層関係定義と実際のレコードの継がりを示している．リングとはこのように親レーコドであるAと子レコードであるBとを関係づけるものであり，最終的にはAに戻る**輪**の形をなす．リングにおける実際のレコード結合は，それぞれのレコードに持たせたポインター，つまり相手レコードのデータベース上の格納アドレスによって行なう．ポインターにはリングに沿って正順の方向である NEXT 逆順の PRIOR，各子レコードから直接親レコードを示す MASTER の3種類があり，NEXT 以外は各階層関係ごとに選択可能なオプションである．図1では NEXT 方向のみのポインターによって結合されていることを例示している．またあるリングに結合される子レコードをあるキーによって正順に並べるソーテェドリングの指定もできる．リングを用いれば，レコードの物理的格納位置を知る必要はなく，レコード間の論理関係だけを意識すればよい．図1は，リングを用いた論理構造の基本型であるが，多くのデータ種，レコードタイプをその論理関係によってリング付けすることにより，自由な論理構造を定義することができる．図2は川崎重工業（株）船舶事業本部における生産管理データベースの論理構造の一部であるが，このようにリングを用いてネットワーク的構造を定義できる．また図2におけるブロックレコードタイプのように複数種類の親子関係を持つことを**クロスリファレンス**と呼び，データの重複が避けられ，メンテナンスが簡潔になるといった利点がある．リングに沿って検索を行なうことは，特定のデータをランダムに検索する場合効率的でない．このため RAPID ではランダムエントリー機能を用意しており，レコードの指定フィールド値により格納位置を算出し，特定のデータを直接検索することを可能としている．同一データベースに対し，複数キーによるランダムな検索が可能である．

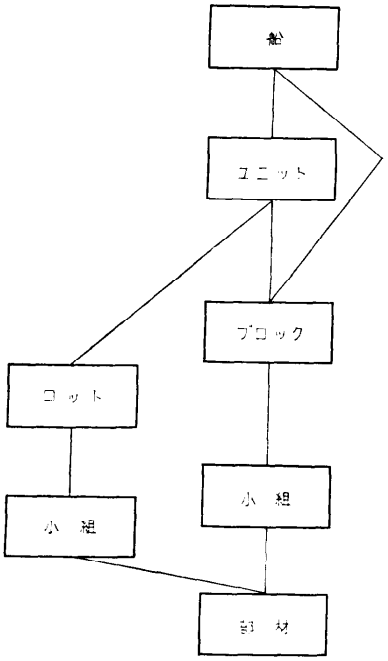


図 2　階層構造例

（2） 記憶構造

記憶構造とはレコードが記憶媒体上に物理的に格納されている形態をいう．データベースは使用者指定の大きさのブロックに分割される．このブロックを**ページ**と呼び，レコードタイプがデータベースの論理的構成体に対して，物理的構成体とすることができる．図3におけるページのデータ領域にさまざまな形態で実際のレコードを格納することができる．1つのページにいくつかのレコードが格納されるが，レコードの種類は1種類でなくてもかまわず，種類の組合せは業務の性質，優先度によって使用者が自由に設定できる．RAPID のアクセス単位はページであり，このページアクセスが最小となるようにレコードの物理的位置を決定することが，記憶構造設計の基本目的である．常

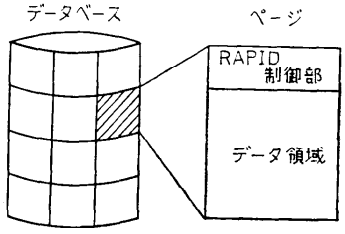


図 3　データベースとページ

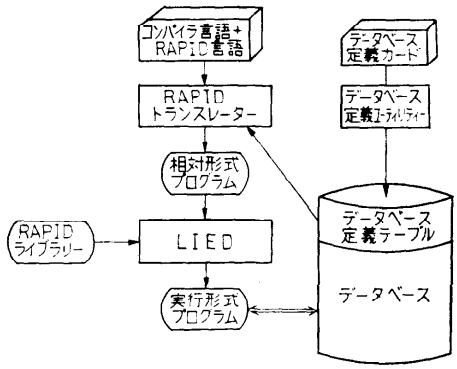


図 4　RAPID のシステムフロー

に定った階層性にあるレコードタイプ群が処理対象となるのであれば，同一ページ群にそれらの複数レコードタイプをまとめて格納する．また，個別に処理されることが多いレコードタイプは単独にページ群を割り当てる，といった配慮が必要である．

2.3 RAPID のシステム

図4はデータベースの定義，RAPID 言語を用いたアプリケーションプログラムの処理過程を示したものである．

（1） データベースの定義

データベース作成の前準備として，データベースのためのスペースを確保し，ページ単位に分割し，そして各ページにページヘッダーを作成するなどいわゆる初期設定を行なう．同時にデータベースの論理的構成，すなわち，レコードタイプの名称性質，およびそれらレコードタイプ間の階層関係（具体的にはリングの性質）を記述した定義カードに従ってデータベース定義テーブルを作成する．これら前処理のためのユーティリティーは RAPID のシステムに用意されている．

（2） アプリケーションプログラムの作成

RAPID はホスト言語方式をとっているので，使用者はコンパイラー言語（COBOL，FORTRAN，ALGOL）の中で，RAPID コマンドを自由に用い，データベースの作成，更新，検索処理を行なうアプリケーションプログラムを作成できる．RAPID 言語とホスト言語で作成されたプログラムは，各ホスト言語ごとに用意されているプレコンパイラーにより翻訳される．プレコンパイラー方式の利点はプレコンパイル時に文法的，論理的エラーの検出ができ，デバッグが容易であるとともに，コンパイラーレベルでデータベースを容易に処理することができる点にある．RAPID コマンドは表1に示す通りである．

表 1　RAPID コマンド

種別	RAPID コマンド定型
制御系コマンド	RPD-OPEN　FD 名
	RPD-CLOSE
	SET CURRENT　レコードタイプ名
	WHEN { ERROR / END / MISS / WARN / ENDLIST / NOTONLY / FREE / NOSPACE } GO TO　手続名
	STATISTICS { PRINT / NOPRINT } 〔TOTAL〕
	MACRO　登録名
GET系コマンド	GET { NEXT / MASTER / PRIOR } リング名
	GET CURRENT
	GET DIRECT
	GET RANDOM　レコードタイプ名 〔DIRECT〕
	CONTINUE
PUT系コマンド	DELETE　レコードタイプ名 〔{ALL / ONLY}〕
	SET FREE　リング名
	MODIFY　レコードタイプ名
	CONNECT　リング名
	PUT レコードタイプ名1 { DIRECT / FROM { ページナンバー1 TO ページナンバー2 / * } / NEARLY　レコードタイプ名2 } 〔INT = { 整数値 / * }〕
	PUT RANDOM　レコードタイプ名 〔DIRECT〕 〔INT = { 整数値 / * }〕

（3） データベース処理の流れ

ユーザーズプログラムにはトランスレータによって，レコード作業領域，RAPID コマンドとのコミュニケーションエリア，およびページバッファが展開され，さらに LIED 時にそのプログラムで使用する RAPID コマンドサブルーチン† が自動的に組込まれる．

これらユーザーズプログラム，各 RAPID 領域，RAPID コマンドサブルーチン，およびデータベース間の制御の流れ，データの流れを示したのが図5であ

† コマンドサブルーチンの組込みにはこのようにジョブ毎に組込む方法と，多重処理に有効なリエントラント形式の共通モジュールとしてシステムに組込む方法とがある．FACOM 230-45S/55 用 RAPID，および FACOM 230-60/75 用 FMS ではコマンドの性質によって，それぞれシステム組込みとするか，ジョブ組込みとするか選択できるようになっている．

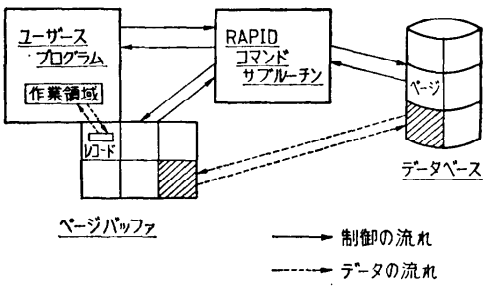


図 5 データベース処理の流れ

る.

例えばユーザーズプログラムで，あるレコードに対して GET を出すと，実際の動作は RAPID コマンドサブルーチン（GET サブルーチン）が行なう．該当レコードの存在するページがページバッファ上になければ，そのページをデータベースからページバッファに読み込む．さらに該当レコードだけをレコード作業領域に転送する．ここでユーザーズプログラムに制御が戻り，このレコードを自由に使用して，処理を続けることができる．

このようにユーザーズプログラムではデータベース処理を意識することなく，コンパイラーの中で，簡単な RAPID コマンドを記述することにより，データを自由に取り出したり，格納することができる．

2.4 運用機能

RAPID は，データベースの効果的な運用をはかるため，次のような機能をもっている．

（1） データ保護機能

RAPID は変更を許すレコードタイプと変更を許さないレコードタイプの定義に従って管理している．

（2） リカバリー機能

ハード，ソフトの障害に対して，データベースを迅速かつ，正確に復元するためのリカバリー機能を持っている．

（3） サポート機能

データベース運用のために必要なデータベース定義，統計情報の収集，データベースダンプ，再編成などのコーティリティーが用意されている．

3. データベースの適用とその評価

3.1 適用目的と開発経過

（1） システムの変化とコンピューター技術の導入

川崎重工業（株）の造船部門における船殻工程の生産管理システム **COSPAC**† は，生産工程（主として船殻の内業加工，小組立，大組立，搭載）を近代的な生産システムへ体質改善（作業の標準化，タクトシステム化）を図るため導入された総合的な生産管理（大日程，中日程，小日程計画）のためのコンピューターオリエントなシステムである．昭和 42 年初めから，プログラム開発が始まり，現在小日程レベルの一部のプログラムを残し開発を終り，各ステージでの工程計画に使用されている．その開発当初は，工程の改善が順次実施されるごとに，その工程の管理者のニーズに対応した専用プログラムを作ってきたので，

- ○ システム変更に弱く，
- ○ データファイルも各アプリケーションの単一機能的な要求に対応して作成された専用ファイルであり，
- ○ 同じデータが重複して存在し，
- ○ ファイルが多量となり集中的な管理が難しい

等の欠点を持っていた．この時点では，作業者レベルのかつ単一ステージという狭い範囲の情報のみを扱えばよかったが，全体の体系が整備されるにつれ，管理者レベルの要求する縦横の複雑な関連を持った，より高度な情報が要求されるようになり，それと共にシステム自体も単一機能的なものではなく，管理者の意思決定に使用するシュミレーション的なものへと変っていった．このような利用者のシステムに対応して，プログラム担当としては，総合的生産管理システムにふさわしい新技術を導入する必要にせまられた．このため昭和 44 年からプログラムシステムの再設計が検討され，

- ○ 全体のプログラム開発を効果的に行なうためにプログラムのモジュール化，
- ○ システムデータの集中的データベース化

の研究と実施が決定された．プログラムのモジュール化と集中的データベース化とは，あい補ってプログラムの生産性や，処理の効率性を促進する有力な手段を提供するものである．前者はプログラムを機能ごとに（例えば，INPUT 処理，OUTPUT 処理，山積や平準化の各種アルゴリズム）分解し，新たなアプリケーションの要求に対応させるものであり，後者は必要なデータを集中的にかつ有機的な関連のもとに管理することによって，RAPID の項で述べたような適用メリットを狙うものである．

（2） データベースの開発経過

生産管理における多種多様な情報要求に応じられる

† Computer System for Planning And Control

ような汎用データベースを開発するには，かなりの人と時間の投資が必要である．従って，いきなり全体のシステムを再検討し，大規模なデータベースシステムを作成することに危惧を感じ，まず第1ステップとしてパイロットデータベースを作成し，実験，調査を行なうこととした．幸い昭和45年に計算機が FACOM 230-50 から FACOM 230-60 へ更改されることに決定し，データベースを取り扱うソフトウェアとして RAPID が提供されることになった．このパイロットデータベースの目的は，

○ 論理構造，記憶構造が生産管理のアプリケーションにふさわしいものであるか，
○ 処理効率はどうか，
○ RAPID の使用性はどうか，
○ 担当者としてのデータベースの理解と技術の向上

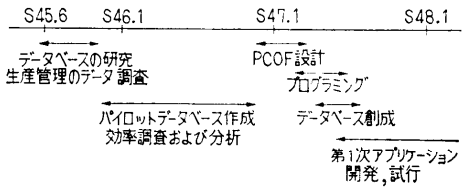
である．その結果，RAPID で扱うネットワーク構造が COSPAC のアプリケーションで使用し得ると判断し，その実験で得た経験を基に，論理構造の変更，記憶構造の変更等を行ない，さらに適用範囲も拡げ，資材システムのうち COSPAC と直接関係のある鋼材システムとの結合へと拡張し，データベース† 中心の COSPAC システムの実用化を図った．開発経過は表2の通りである．

3.2 適用分野

(1) 適用業務

現在の COSPAC が対象としているのは造船における船殻工程の一貫した生産計画システムである．造船は，個別受注生産であり，工事自体が大型で工程期間も長い．しかも一般には同一仕様ではなく，材料部品も多種多様で，受注後初めて手配されるものがほとんどである．また，部品や材料を現場に持込んでから加工，組立を行ない，下請工場や外注工場の利用度も高く，工程の構成も複雑となり，物を厳密に管理することに加えて納期を確保（工程を keep）することが重要となっている．従って造船の管理システムは，

表 2 開発経過



† PCOF (Production Control File) と称する．

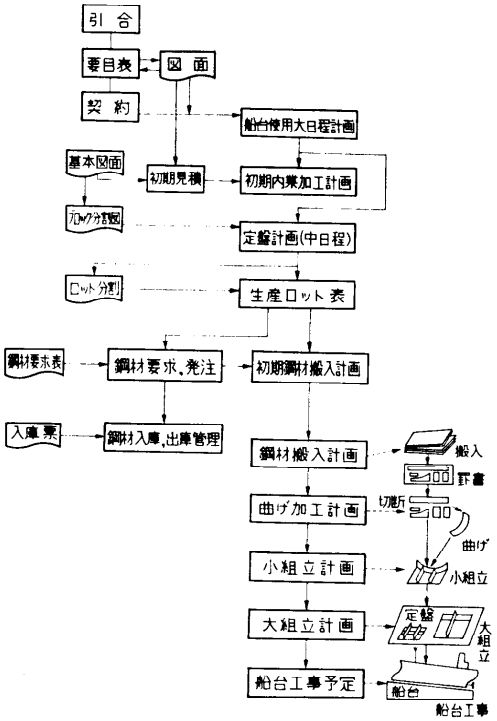


図 6 適用業務図

○ 受注に当って正確なコストの予測，
○ 全体の正確な日程の予測，
○ 受注後は，決定されたコストおよび日程をいかに守っていくか

などがポイントとして挙げられる．このうちCOSPACでは，特に工程の計画を注目し，図6で示すように各ステージでの日程計画を中心としたサブシステムで構成されている．

船の受注が決ると，契約に基き，大まかなデータが見積られ，他の船との関係，作業能力との関係で大日程計画が立てられる．基本設計，詳細な生産設計と進むにつれ，中日程計画，さらには小日程レベルへと展開される．図6で示しているように中日程の段階で，船の構成品であるブロック（大組）が決められ（ブロック分割図），船殻のネック工程である定盤計画（ブロックの組立中日程計画）が立てられる．そのブロック分割図に従い，素材である鋼材が決まり，鋼材のグループとしてのロット割りが決められる．定盤計画に基き，逆に，ロットの現場への搬入予定が立てられる．それを生産ロット表と称しているが，これは，定盤計画と共にマスタープランとなっている．搬入された鋼材は，罫書され，切断，曲げそして小組立という

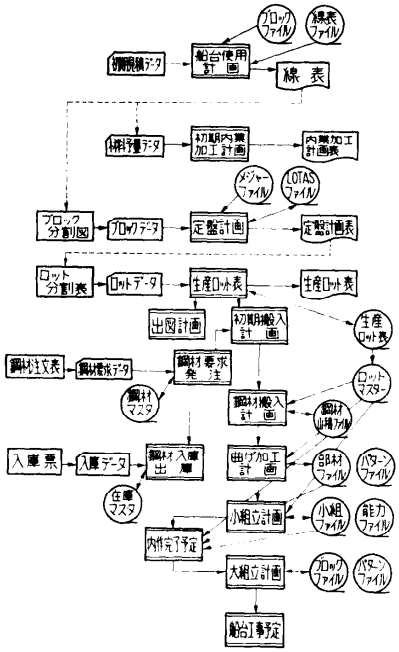


図 7　旧 COSPAC システム図

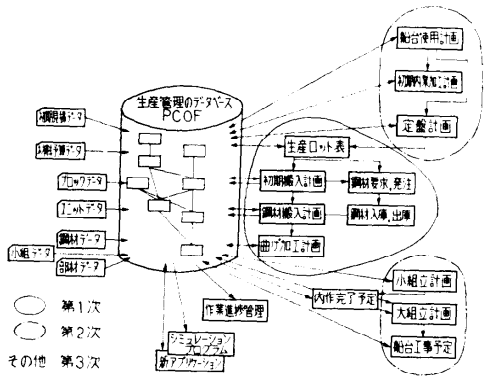


図 8　新 COSPAC システム図

内業工程を通り，外業工程へ引渡される．そして，大組立工程でブロックとなり，船台上で船の型に形作られていく．COSPAC のアプリケーションは，これらの各ステージの工程計画を行なっている．

（2） 業務とファイルの関連

COSPAC システムは，従来図7で示すように1つのアプリケーションごとに，クローズされた専用ファイルを持っていた．そのため，ファイルが多量で，データも多数のファイルに重複している．図8の新システムは，データベース中心に各アプリケーションがそれぞれ関係づけられて構成されている．現在第1次の内業関係のアプリケーションを試行中であるが，第2次で外業関係のアプリケーションを開発し，第3次では，従来のシステムでは考慮されていない総合管理的なシステムへと発展させる意向である．

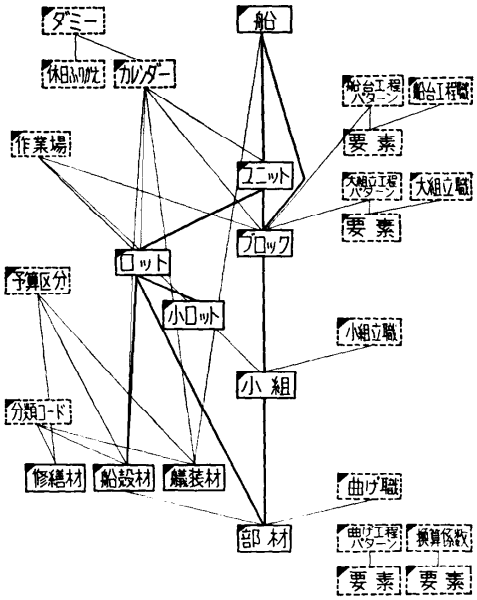


図 9　PCOF データ構造

3.3 データベースの構造

（1） 論理構造

生産管理で扱われるデータは，大きく分けて

- ○ 物に関するもの（いわゆるトランザクションデータなるもので船の構成品），
- ○ 工程に関するもの（トランザクションがたどる過程を示す），
- ○ 作業場に関するもの（トランザクションを扱いうる能力等），
- ○ 時間に関するもの（カレンダー），
- ○ その他マスター的データ

に分類できる．

これらのデータを生産管理の現在あるいは将来考え得るあらゆるアプリケーションでの処理を考えて，相互に有機的に関連づけを行なって，図9のような論理構造を決定した．

物に関するデータは，図9の実線で囲んだレコードタイプで示しているが，船という製品が鋼材から，部材，小組，ブロックと組立てられていく過程が示され，生産工程での管理単位であるロットまたはユニッ

トという単位でも関連づけを行なって，各ステージ間のトランザクションの関係を明確化している．その物のまわりをとりまいている種々のデータ（図9において点線で示している）は，生産管理の各アプリケーションで，例えば，

○ Aブロックが，どの定盤で，どのような組立工程をたどるのか？
○ M月D日からE日までに，搬入予定のロットは何か？
○ 未入庫の船殻材は何か

といった要求があるが，そのような要求に応じられるようなエントリーを考えて構成されている．

（2） 物理構造

PCOF の設計に際して，前述の論理構造と共に注意を払ったのは，実際にデータを記憶媒体上のどこに格納するかということである．**2.2** で述べたようにデバイス上のどのページのどこに，どのレコードを格納するかという記憶構造は，レコードのアクセススピード，データベースのスペース効率等に大きく関係する重要な問題である．

パイロットデータベースの実験において，アプリケーションのデータ検索の性質を詳細に分析し，検索時間やアクセス回数等の統計情報をもとに検索効率を計算し，また，データ創成の単位，削除の単位，タイミング等も考慮し，物理構造を決定した．

まず第1にページサイズの決定である．PCOF は，第1次では F472K† 1台である．RAPID の Read/Write の単位であるページサイズは，レコードの大きさ，すなわち，格納のスペース効率と，レコードのアクセスの順序，すなわち，検索効率を考慮する必要がある．

PCOF では，レコードの最大長が 63 ワード†† と大きいこと，また，1つのレコードを検索したら，その近くのレコードを連続して検索される処理が多いことなどから，ページサイズは 3519 バイトを選び，最大ページナンバーは 7830 ページとした．

次に記憶構造を図 10 のように設定したが，その決定にあたり以下のようなことを考慮した．

○ COSPAC では，一旦あるエントリーからレコードが検索されるとそのまわりのデータは連続してアクセスされることが多い．
○ トランザクションデータ（船の構成品）は，生産設計の各レベルで，船番単位に，しかも各ステージごと（すなわちレコードタイプごと）に作り出される．
○ 進水（船台工事の終り）または，引渡しの時点でデータは不要となることが多い．

† 容量：28M バイト
†† 1ワード＝4バイト＝36 ビット

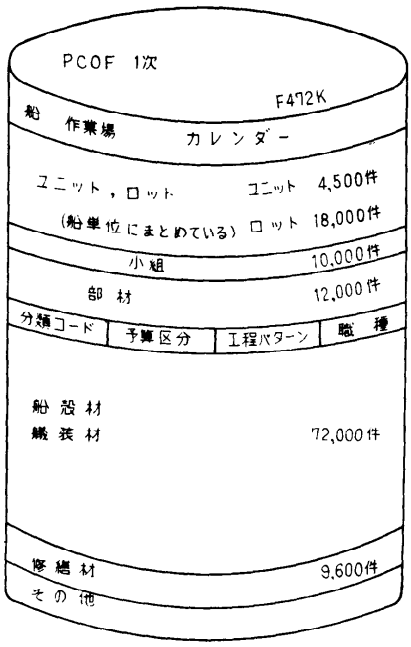


図 10 PCOF の物理構造

船単位でデータの発生，消滅が起こるのでスペース管理も考慮している．

3.4 評価

昭和47年5月末に，PCOF の基本ルーチン，基本アプリケーションの開発を終り，その後アプリケーションの追加開発を進めながら，PCOF データベースの試行運用を行なっている．たまたま，既存のシステムに適用したので，開発効率上では，かなり問題があったが，専用ファイルをもった既存システムとデータベース中心の新システムを並行 RUN させて比較できるという結果となった．これを比較すると次の通りである．

（1） 集約度

旧システムでは，順編成ファイル中心であるため，多種多様な情報を必要とするアプリケーションでは，多数のファイルを必要とし，それらのファイル中のデータ相互の関連づけをプログラムで行なうため，プログラムが複雑となる．また，エントリーごとに SORT

を行なったりするためジョブステップも増える．

他方，新システムでは，唯一のファイルをさまざまなエントリーからアクセスできるので，簡単なジョブで処理できる．新旧両システムのジョブステップ数とファイル数の具体的な数値例を表3に示す．

表3 集約度

アプリケーション	処理	ジョブステップ 旧	ジョブステップ 新	ファイル数 旧	ファイル数 新
ロット表	新規登録	2	2	2	1
	通常変更	3	2		
	強制変更	4	2		
鋼材搬入計画	通常処理	11	2	3	
	ファイルメンテナンス	2	1		

（2） データの重複度

アプリケーションごとにクローズされた専用ファイルを持った旧システムでは，同じデータが複数のファイルに存在している．新システムでは，データに関連づけを行なって，無駄な重複を避けるとこができた．参考までに重複度がどの位減ったかを一例をとって表4に示している．この表では，データベース上のデータを1とした場合のロットファイルと鋼材ファイルのデータ数を示している．この他にも，最も悪い例ではあるが，旧システムでは，同じデータファイルがフォーマットを少し異にしただけで存在するといった例もある．

表4 データ重複度

データ	旧 ロットファイル	旧 鋼材ファイル	新データベース
船	500	2,000	1
ユニット	3	12	1
ロット	1	4	1

（3） 処理効率

ある1つの場面のみをいれば，汎用的なデータファイルより専用ファイルの方が効率がよくなるのは当然であるが，処理効率の判断の視点をどこに置くかは，そのアプリケーションの処理系体，使用頻度や業務の優先度等によって違ってくる．参考までに旧システムと比較できる新システムの一部について，その時間比を表5に示す．

（4） プログラム開発工数

プログラムの開発工数は，データベースの構造に対して簡単な知識を持った人であれば，専用ファイルをいくつかアクセスし，マッチングを行なったり，データの関連づけをプログラムで行なう処理をするより，簡単な記述で，自由に必要なデータのみをアクセスできるデータベースシステムの方が少くて済むのは当然である．また，プログラミングコストの 60% 以上を占めているといわれるメンテナンスに対して，データベースシステムは，その真価を発揮する．例えば，レポート項目の追加や新しいレポートの要求等があると，従来では，レポート用の中間ファイルの変更，マスターファイルの変更あるいは新ファイルの作成を迫られたが，データベースシステムではその必要がない．さらに，ここでさきに述べたプログラムのモジュール化がその効果を現わす．上の例で考えると，モジュール化されていれば，データベースの情報を取り出して，レポートを作成するプログラムを追加，作成するか，あるいは修正するというだけでよい．データの取り出しについても，プログラムにしても他のことを考えずに，関連ある部分のみ考慮すればよいので，メンテナンスコストは少く済む．

表5 処理効率

アプリケーション	処理	時間比 旧	時間比 新 CPU	時間比 新処理時間
ロット表	新規登録	1	0.88	2.48
	通常変更	1	0.42	0.78
	強制変更	1	0.34	0.83
鋼材搬入計画	通常処理	1	0.34	0.46
	ファイルメンテナンス	1	1.33	1.23

（5） 運用評価

PCOF データベースで問題となったのは，運用管理である．ある1つのジョブを眺めれば，ファイルは常に同じものを処理すればよいし，ジョブステップも少くなってオペレーション自体は確かに簡略化され楽になっている．データベースシステムに不慣れのためのトラブルもあるが，主な問題点を以下に列挙してみたい．

○ バックアップに関する問題

旧システムでは，アプリケーションごとに重要ファイルについてタイミングを考え，バックアップ MT をとるということで充分であったが，データベースシステムでは，唯一のファイルを多くのアプリケーションが，検索，更新を行なうので，バックアップ MT をとるタイミングが非常に難しい．(オンラインデータベースの場合はジャーナル MT を自動的にとる配慮をするが，PCOF では，データの出入り頻度が少いのでこの方法は

とっていない.) また，テスト用のダミーデータベースを簡単に作成できないのでデバッグジョブもその唯一のファイルを使用するので非常に危険を併う.

○ 再編成に関する問題

RAPID のようなデータベースでは，本来の意味での再編成が非常に難しく，スペースの管理にかなりの労力を必要とし，データベース上の格納状態を常に意識する必要がある.

○ データベース管理者の必要性

PCOF のように，バッチ処理のジョブのみで，データの機密保護もそれほど必要とされないデータベースであっても，上記の問題から，何らかの型でデータベースを管理する人が必要である．データベース管理者は，データベースが，今どのような状態であるかを常に把握している必要性がある.

4. まとめ

以上，RAPID の適用例として，PCOF の開発について述べてきた．川崎重工業（株）船舶事業本部ではこの生産管理のデータベースをひとつの足がかりとして，経営，資材，生産の管理システム全体へと発展させ，理論の展開に終始することなく，実際に試行しながら，データベース中心のトータルシステムの確立を図っている.

最後にデータベースシステムの設計，開発にあたって，最も注意を要した点などについて述べ，しめくくりとしたい.

PCOF の開発目的で述べた通り，データベース適用の目的は，情報の統括管理であり，多種多様な業務による共同利用である．しかし，このような完全に汎用性を持ったデータベースの設計は非常に難しい．本来，ファイルの汎用化と処理効率は相反するものであり，この矛盾をいかに克服するかは，全てデータ構造の設計にかかっている．例えば，拡張業務に対する配慮が論理構造設計時になされていなければ，柔軟性のあるデータベースシステムとはいえないし，また各業務の性質，データの関連によって物理構造の設計が行なわれていなければ，処理効率に大きく影響する．このようにデータ構造の設計はデータベースシステムの死命を制する作業であり，全適用業務，全データの性質，関係の詳細な分析が必要である．しかし，データベースの設計には試行錯誤の要素が多いのも事実であり，この点からも，パイロットデータベースの設定，評価の作業はデータベースシステム設計上のひとつの手段として有効と思われる．従来，データベースの必要性が叫ばれながら，導入が敬遠されがちであったのもこのデータ構造設計の困難さと，処理効率に対する不安であったと考えられる．しかし，専用ファイルに依存する従来のシステムでは，情報の多様化，増大に対処できなくなるのは必然である．PCOF の実績評価における情報管理の簡潔化，業務要求に対する柔軟性に注目すれば，将来展望において，データベースシステムの開発メリットは明白である．このことからもデータベース化への第1歩は，システムの将来構想に対して不可欠と思われる.

以上，データベース開発の具体例により，データベース適用の手順，メリット，さらに問題点についてご理解いただければ幸いである.

参考文献

1) 日本造船学会論文集：造船の生産管理におけるコンピューターの適用，
第1報，124号，1968年.
栗岡辰己，野間口幸宏（川崎重工業神戸工場）
第2報 125号，1969年.
栗岡辰己，竹内秀司，藤田牧男（川崎重工業神戸工場）
第3報，127号，1970年.
栗岡辰己，谷本信身，野間口幸宏（川崎重工業神戸工場）
2) FACOM 230-60 RAPID 解説書.
3) FACOM 230-60 RAPID プログラミング解説書.

（昭和48年3月16日受付）