高松短期大学紀要，20．1～10（1990）

学事情報システムにおけるデータベース構築の取組

佃　昌道

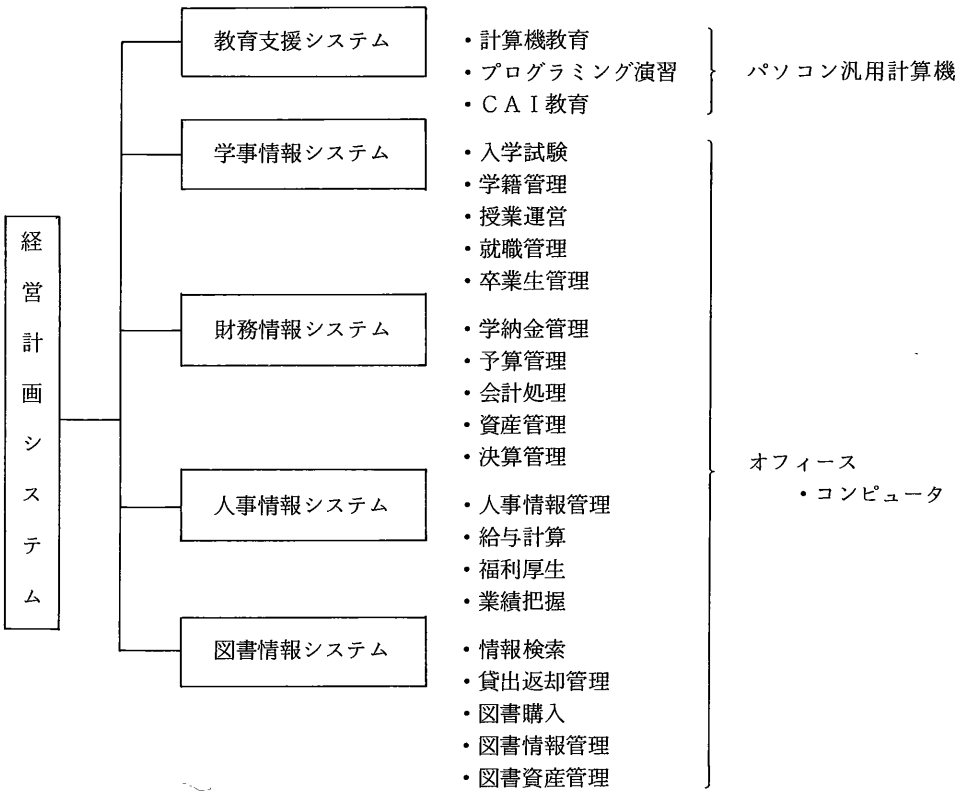
はじめに

学校事務におけるコンピューター化が進むにつれ，利用部門の要求は増大し，データ・ファイルの数は増加の一途をたどっている。また，非定型業務に対して計算機を利用する機会も増加している。

このような現状に対応するためには，蓄積された情報をより効果的に利用することが重要である。本稿では学事情報システムを中心に，現状と対比しながらデータベース化の方法を述べていく。

Ⅰ．学校法人システムの概要

本学では，昭和58年より各システム毎に開発を続け，問題をかかえながらも理想システムに近づきつつある。本学における経営計画システムの理想図を図１に示す。



（図１）

Ⅱ．学事情報システムの現状

理想システム中の学事情報システムと本学での対象業務，主要ファイルの関係を表１に表す。なお表中の下線部の項目は現在開発中のシステムである。

対象業務	理想システム	具体的業務内容	主要ファイル
入学試験	入学試験	学校情報 願書処理 入学試験	願書マスタ 点数マスタ 高校住所ファイル 合格者ファイル
教　　務	学籍管理	学籍移動 各種証明書 学籍簿	学籍マスタ 成績ファイル 研究室マスタ
	成績管理	成績証明書 卒業判定 各種免許状	評価ファイル 成績ファイル 科目マスタ
	授業運営	カリキュラム管理 時間割作成 履修登録	科目マスタ 履修ファイル 教員マスタ
	開放講座運営	開放講座案内 開放講座処理	
就　　職	就職管理	企業情報管理 就職実績管理 就職企業検索 教育実習	企業ファイル 学籍マスタ 卒業生マスタ
同 窓 会	卒業生管理	卒業生情報管理 同窓会名簿	卒業生マスタ 企業ファイル

（表１）

Ⅲ．現状の問題点とデータベースの利点

1.　現状の問題点

表１の主要ファイルを見ると，ファイルの数も少なく，データ構造も単純化されているように見えるが，現実には複数の中間ファイル，バックアップ・ファイルやコントロール・ファイルが作成され，現在学事情報システムだけで，60本から80本のファイルが存在している。このように，ファイルの数が増加するに従い，さまざまな問題が生じてきた。

まず第一に，必要な項目をすべて含んだファイルのない場合が多いので，目的にあったファイルを作成しなければならず，この事により同じタイプのデータが多くの場所に記憶され，ディスク容量の増加の原因となる。たとえば，成績記入表を出力する際に履修表のファイルを展開し，学籍番号と科目コードだけのファイルを作成する。そして，そのファイルを基に学生の名前と授業科目名を検索し出力することが必要になり，４種類のデータ項目が２ケ所に存在することになる。また、同じデータ項目が，複数ファイルに存在すると更新時点の相違によってはデータの値が異なってくるおそれがある。先程の例でいうと，履修表の展開を

行った後に履修表のファイルのデータ更新を行うと，出力結果との相違が起こるのである。

第二に，非定型な情報の要求に対し，さまざまなファイルの中から必要な項目を抽出し，その条件にあったように変換しなければならないため，即座に回答が得られない。また，そのことがバックログの原因になり，基幹システムの開発のさまたげになる。

第三に，ファイルの項目の追加や変更を行うさいには，そのファイルを利用しているアプリケーション・プログラムを全て変更しなければいけないために大変な時間が費やされる。このため，大幅なシステム変更ができにくく，エンド・ユーザからの不満を解消できない場合がある。

2. データベースの利点

前項で，従来のファイルを中心とした情報管理手法の問題点が理解されるであろう。

データベースはデータをプログラムから独立させることにより，データ記述はアプリケーション・プログラムと切り離された形で記憶される。

このことにより，データ項目の追加・変更が容易に行え，またデータの重複が無いためデータの独立性を保つことができる。

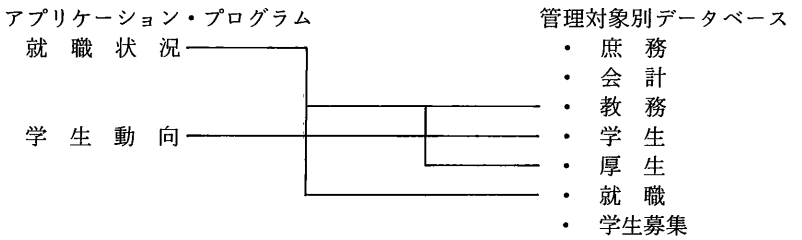
もう一つの特徴は，データベース検索ソフトウェアが用意されていることである。これによって，利用部門はプログラムを作成せずに，コンピュータのデータの取り出しや，簡単な報告書の作成が会話方式で利用できる。

このような，データベースの特徴を利用することにより，データの共有化，プログラムの生産性の向上，データの一元管理，データ更新時の矛盾の削減等の効果を得ることができる。

Ⅳ．データベースの概要

1. データ環境

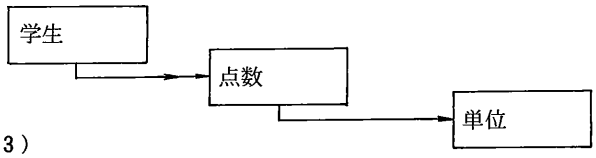
データベースを設計する場合，管理対象を中心に考える場合とアプリケーション・プログラムを中心に考える場合とがある。アプリケーション・データベースによるデータベースの設計は各アプリケーション毎にデータベースを設計するためデータベースがたくさん存在し過度の重複を引き起こしやすくまた，データベースシステムの主要な利点を達成出来にくい。それに対して，管理データベースは組織の管理対象毎にデータベースを構築するためアプリケーションからの独立性が高い，データベースの数も少なくて済む。（図２参照）本項では，管理対象データベースを中心に考えゆく。



（図２）

2. 階層型レコード構造

データベース管理システムはあらゆる論理データ構造を表現できる必要がある。例えば「学生」というレコードがありそのレコードに関連する複数の「点数」というレコードがあるとしよう。「学生」と「点数」のレコードの関連は次のように描くことができる。

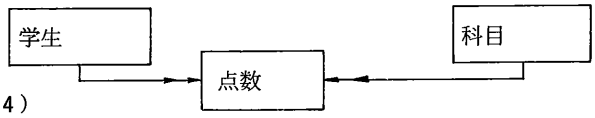


（図３）

「点数」レコードに入る→は各々の「学生」に対して多数の「点数」レコードがあることを示している。学生を見てその点数を見た上，点数の科目を調べたいとしよう。すると図３のようなデータ構造が必要となる。この構造をデータと階層型構造という。階層中の個々のレコードは上位レベルの階層の１つのレコードと関連している。

- プレックス構造

レコード間の関連が厳密に階層的でない場合もある。例えば科目に対してどのような点数があるかを示す「科目」レコードを加えたりすることがある。



（図４）

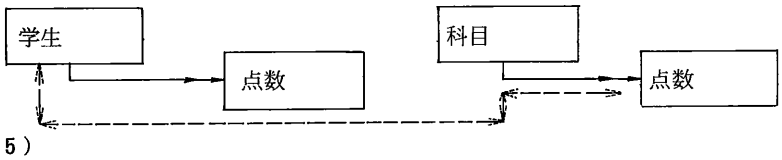
図４のように，階層型構造より複雑なレコード群の構造はプレックス（網）構造と呼ばれる。我々が表現する必要のあるデータの多くはこのプレックス構造である。

3. データベース構造の表現

ＤＢＭＳはレコード間の連関を表現できなくてはならない。この表現方法には種々の方法があるが，このような構造を表現するに主要なアプローチには階層型，ネットワーク型，関係型，の３種類のアプローチがある。

図のプレックス構造を３つのアプローチで示したものが図５～図７である。

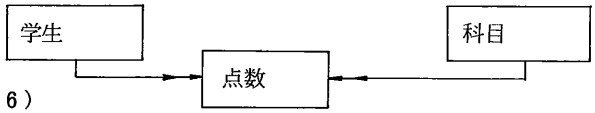
階層型のアプローチではプレックス構造を解体して階層にする。階層を結び付けるのにポインタを使用する図５の点線の矢印がこのポインタを表している。レコード階層は単純にレコードをディスクなどに配列する順序によって表すことができる。したがって一つのレコードは階層の中で関連するレコードと物理的に近接している。レコード及び関連レコードは共に即座にアクセスすることができる。しかし，一つの階層から他の階層に移動する場合に時間がかかる。



（図５）

ネットワーク型アプローチはレコードはセットと呼ばれる２レベルの階層にグループ化さ

れる。セットは重ね合わせてネットワークを形作ることができる。つまり，１つのレコードは複数のセットに対してその部分となることができる。一つのセットの中での関連レコードはそれらが一緒に読み込めるように物理的に近接した位置に記憶させることができる。



（図６）

データ構造を表現する第三の方法は関係型アプローチである。前にあげた二つのようなポインタ構造を避け，あらゆるデータを関連のないレコードとして表現する。その代わりに，必要な関連を可能にするデータ項目がレコードの中に含まれている。このアプローチでは数学的に記述しやすいデータの単純な表構造ができる。レコードの関連を扱うため，余分なデータ項目がレコードの中に幾つか入っている。学生の点数表を印刷する際に学生の名前を求めるために「点数」レコードから「学生」レコードへと渡り歩く必要がある。これをするために「学籍番号」というデータ項目が「点数」レコードの中に入っている。つまり，関係型データベースでは連結ポインタが存在しない代わりにデータ項目に索引を持たせることができる。

学　生

学生番号	氏名

点　数

科目番号	学生番号	点数

科　目

科目名	単位	

（図７）

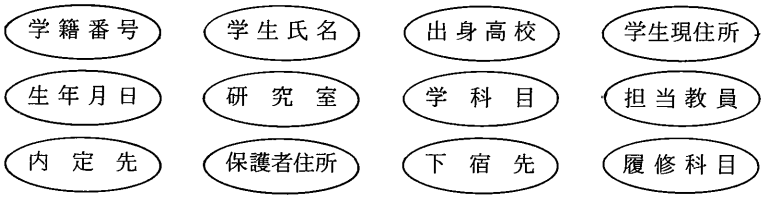
4. データベースの論理的設計

データベースの全体的な論理設計を行うことがデータベース環境を成功させる重要な要因である。またこの仕事は利用部門の支援が必要となる。

利用部門との意思疎通をはかるためには，データ構造図を用いる。この中で泡状チャートはデータ特性を簡単に表現できるために，データの表現や検討がわかりやすい。以下に簡単にその説明を行う。

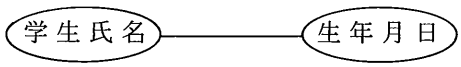
データの基本要素をデータ項目といい利用者によってこれ以上細かくできない，意味のあるデータタイプのことをいう。

このデータ項目の各々のタイプをひとつの楕円（泡）で表現する。



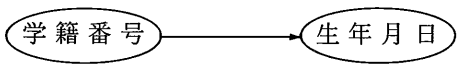
このデータ項目とデータ項目間に関連がつけられる意味をもってくる。たとえば，学生氏

名と生年月日が結び付けられると意味をもってくる。



したがって，データベースはデータ項目だけでなく，データ項目間の連関と共に構成される。いろいろな種類のデータ項目が大量にあり，そのようなデータ項目がどのように関連するかを表す展開表が必要となる。この展開表をデータモデルと呼ぶ。データ項目間を結ぶ線には単一矢線と２重矢線の二種類がある。

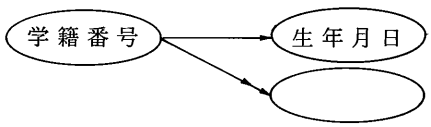
たとえば，「学籍番号」のある値に対して「生年月日」の値は１つしかない。したがって「学籍番号」から「生年月日」に単一矢線を引く。



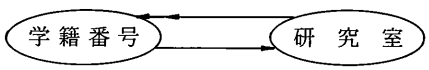
この場合「学籍番号」は「生年月日」を識別する。すなわち，「学籍番号」が分かれば学生の「生年月日」が分かるということである。

これに対し，「学籍番号」の値に対し「各科目の点数」の値は（０個，１個を含み）複数の値をもつ場合がある。この時「学籍番号」から「科目の点数」に２重矢線を引く。

いままでの２つの状況を泡状チャートで表すと次のようになる。



また，２つのデータベースにおいて，両方向の関連があり，前向きと逆向きを含めて４つの組み合わせがある。たとえば，学籍番号と研究室を考えてみよう。研究室から学籍番号への逆向き連関は必要である。その理由は利用者が，どこの研究室に属している学生かを知る必要があるからである。



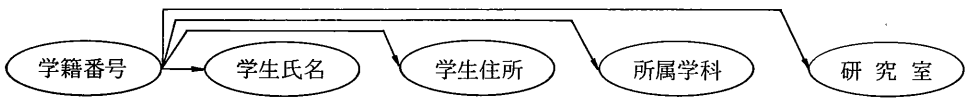
5. データ項目のグループ

Ｎ個のデータ項目間にはＮ（Ｎ－１）個の連関があり得る。データ項目間の連関は膨大な数になるため，すべての連関を取り扱うのは現実ではない。そこでデータ項目をグループ化し，グループ間の連関を検討することによって連関の数をある程度妥当な数に縮小できる。

データ項目グループは次の様に表される。

学籍番号	学生氏名	学生住所	所属学科	研 究 室

このデータ項目グループが泡状チャートでは次のように描く。



Ⅳ．学事情システムにおけるデータモデルの作成

データベースを構築するには，データ構造の安定化や柔軟な変更に対処できる形にベータを分析し，データ項目をレコードやセグメント，タプルにグループ化しなくてはならない。この手法の一つとして第三正規形がある。正規化の手順としては非正規化データの中の繰り返しグループを持つレコードを分割する。このことを第一正規形という。次にキー以外のすべてのデータがキーに従属する形にしそれ以外は分割する。このことによりすべてのデータ項目がキーに関数従属する。このことを第二正規化という。そして最後にキー以外の項目の関係は従属性がないようにレコードを分割する。このことによりキー以外のすべてのデータ項目はキーに完全従属し，なおかつデータ項目間には全く従属関係が無くなる。このことを第三正規化という。

ここでは，学事情報のなかの教務に関する主要ファイルとして，学籍マスタ，成績ファイル，研究室マスタ，評価ファイル，科目マスタ，履修ファイル，教員ファイル，企業ファイルを例に正規化を行ってゆく。

○　学籍マスタ

学生の基本情報として使用される学籍マスタの内容は図のような形になっている。学歴とは，入学してからの学生の状況を記録したもので，学民により様々なものになる。また住所は現住所，親元の住所，下宿先の住所という具合に記録されている。ここで繰り返しグループの削除し主キー（連結キー）全体に従属しない属性の除去を検討し主キー以外のデータ項目に従属する属性を除去すると次のように正規化される。ここで，住所については区分をつけてどの種類の住所かを判断する必要がある。

学籍マスタ（非正規化）

学籍番号	コースコード	研究室コード	入学年月日	卒業年月日	氏名・カナ	氏名・漢字	生年月日	性別	本籍地	学歴		住所				保護者	出身高校名	就職先	内定日
										年月日	内容	郵便番号	住所CD	字・番地	棟・番号	電話番号			

4回繰り返し　　3回繰り返し

学籍マスタ（正規化後）

学籍番号	コースコード	研究室コード	入学年月日	卒業年月日	氏名・カナ	氏名・漢字	生年月日	性別	本籍地	保護者	出身高校名

学生・学歴

学籍番号	学歴	
	年月日	内容

学生・住所

学籍番号	住所					住所区分
	郵便番号	住所CD	字・番地	棟・番号	電話番号	

内定先

学籍番号	就職先	内定日

（図8）

○　履修ファイル

次に学生の履修状況を記録した履修ファイルについて考える。履修ファイルはテーブルの位置に曜日・校時のデータを持っているため，繰り返しグループの削除のさい，曜日・校時のデータを追加する必要がある。

履修ファイル（非正規化）

学籍番号	前期	後期
	科目コード	科目コード

└─各期49回繰り返し

履修ファイル（正規化後）

学籍番号	科目コード	曜日コード	校時コード	学期区分

（図9）

○　評価ファイル

試験が終了すると学生の評価を蓄積する評価ファイルが作成される。本学では，再試験，追試験の処理を行うため，3回の履歴が必要になるため次のようなファイルを構成している。ここで繰り返しグループを削除すると学生の点数，判定が二重に現れるために複雑になる。そこで，必要な項目とその項目の従属性を考えると次のように変わる。

評価ファイル（非正規化）

年度	キー		回数		最終点数	最終判定	学期区分	試験履歴			
	科目コード	学籍番号	授業回数	出席回数				試験区分	点数	判定	試験日

6回繰り返し
試験履歴

評価ファイル（正規化後）

年度	キー		回数		最終点数	最終判定	学期区分
	科目コード	学籍番号	授業回数	出席回数			

キー		回数			
科目コード	学籍番号	試験区分	点数	判定	試験日

評価ファイル

年度	キー		回数		最終点数	最終判定	学期区分	最終試験区分	最終試験日
	科目コード	学籍番号	授業回数	出席回数					

（図10）

○ 成績ファイル

点数の入力が終了すると評価ファイルをもとに学生個人の成績ファイルが作成される。科目の分類別に整理するためにたくさんのテーブルが存在しまた，合計計算まで計算される。

成績ファイル（非正規化）

学籍番号	評価データ						単位合計				
	科目コード	単位	点数	出席判定	修得年度	判定	一般教育	外国語	保健体育	専門教育	合計

128回繰り返し

学生単位数合計（正規化後）

学籍番号	単位合計				
	一般教育	外国語	保健体育	専門教育	合計

学生成績ファイル

学籍番号	評価データ					
	科目コード	単位	点数	出席判定	修得年度	判定

（図11）

○ 科目マスタ

授業科目のデータを記憶させた科目マスタを正規化すると次のようになる。

科目マスタ（非正規化）

キー		科目名	単位数	担当者	開講曜日校時	必須区分
年度	科目コード					

5回繰り返し　11回繰り返し

科目マスタ（正規化後）

キー		科目名	単位数	開講曜日校時
年度	科目コード			

担当者ファイル

キー		担当者
年度	科目コード	

必修判別ファイル

キー		必修区分
年度	科目コード	

（図12）

企業ファイルについては，求人の情報と会社の内容に関する情報とが混在しているためこれを二つに分ける。

企業マスタ（正規化後）

企業コード	企業名	企業住所					代表者名	営業種目	設立
		郵便番号	住所CD	字・番地	棟・番号	電話番号			

求人データ

企業コード	求人情報			
	選考日	応募種別	採用人数	その他

（図13）

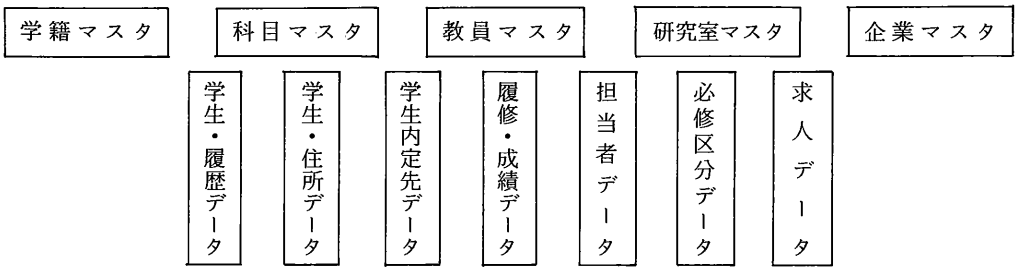
ここでもう少しファイルの内容を考えてみる。学生の科目の履修は必ず評価と結びついてくるため各ファイルの科目コードは学生一人あたりには同一のものとなる。また，必修科目においては卒業までに必ず単位をとらなければならない条件がある。そこで成績関係の各ファイルの重複を取り除くと次のようになる。また，単位の合計は学生毎に常に計算すればよいので，ファイルとして残す必要はない。

履修・成績ファイル

学籍番号	科目コード	曜日コード	校時コード	学期区分	年度	回数		最終点数	最終判定	試験区分	試験日
						授業回数	出席回数				

（図14）

ファイルの形式を変換してまとめると12のファイルにまとめることができる。



（図15）

現在のシステムの８本のファイルは12のグループに変換することができた。ファイルの種類から考えると1.5倍に増えたわけであるが，利用する側からは次の様な利点が上げられる。

履修から成績までを全てまとめることにより，成績の修得状況と履修状況が簡単に検索できる。また，データがリアルタイム科目毎の履修状況や修得状況が即座に検索できる。

学生の履修データの数にこだわる必要がないため，学生の管理データが多数持てる。

その他，データ項目さえ知っていれば非定型に様々なデータの検索が行える。

ここで一つ問題になるのがデータ項目名の定義である。如何にすぐれたデータベースの設計をしても，利用者がその項目がどういう意味であるかが分からなければ，様々な検索を行うことができない。そこで，データ項目の定義と表現について全学的に統一を行う必要がある。そして，できれば計算機の中に標準化されたデータ項目の辞書（データディクショナリ）を構築することによって，より一層データベースは機能を発揮するであろう。

終わりに

学事情報システムの特に教務を中心にデータベースの構築を述べてきたが，今後実際にシステム設設になると処理効率等の問題がある。現在の計算機も1990年４月から機種が変更されるため，本稿で述べたデータベースの作成手順により，息の長いシステムに変更してゆきたい。