

データベース論理設計におけるトランザクションモデリングについて

酒井 博 敬

(株)日立製作所技術研修所

堀 内 一

(株)日立製作所 基盤部

1. はじめに

近年、データモデリング技術の意義は単にデータベース構造設計のため手段としてだけでなく、情報システム全体の構造を決定するための手段としての認識が高まっている。特に企業における長年のシステム開発と保守の繰返しは情報システムの無統制な肥大と内部構造の複雑化を来しており、そのための既存システムの見直し再構築と統合をデータ中心に進めようとする気運が高まっている [NOLAT9], [MART82], [FINK81]。

データを中心にソフトウェアを設計する手法には [JACK76]* [WARN80] がよく知られている。この手の手法は [MYER75] による機能分割を中心としたモジュール化に比して、より客観的なモジュール化基準を提供した。

データを中心にソフトウェア、システムの構造を決定するためには、単にデータの構造的最適化が図られるだけでなく、データが反映する実体や関連の動的特性が明らかにされなければならない。何故ならば、データの確実な更新処理(1次データ処理)はそのデータを利用する向合々処理(2次データ処理)に優先して設計されるべきであり、1次データ処理の設計には実体(関連)の動的特性の考察と、それに基づく一貫性制約やデータ操作の決定が不可欠となるからである。

本稿ではデータモデルを構造部、操作部、一貫性部から構成されるものと捉え [CODD82]、データモデルを記述するERスキーマに、パトリネット記法による動態記述を含める方

法を述べると共に、そのようなERスキーマもつシステムの構造方式について考察する。

2. データモデル化の手順

データモデルを、構造部、データ操作部、一貫性部から構成されるものと捉えると、その設計過程も構造部設計、操作部設計、さらに一貫性部設計の3つの過程に分けて考えることができる(図1参照)

構造部設計とは、データの構造的側面を最適化することであり、その過程は概念レベル、論理レベル、さらに物理レベルに分けられる。概念レベルの構造部設計とは実世界に実体とその関連を捉え、ERモデルとして表現すると共に、その改良、最適化を施すことである。論理レベルの構造部設計は特定のDBMSが提供する論理データ構造に関する最適化を図ることを意味する。物理レベルの構造部設計は特定のハードウェア方式の下で実現されるデータ構造(蓄積構造)に関する最適化の過程をいう。ここではERモデルによる概念レベルの構造部設計を論じる。

一貫性部設計とは、構造部設計によって決定されたデータについて、その完全性、一貫性を維持するために要求される様々な規則や制約を明らかにすることである。一貫性に関する規則や制約は、概念レベルのデータ構造部に関するもの、論理レベルのデータ構造部に関するもの、さらに物理レベルのデータ構造部に関するものに分けられる。これらの規則、制約は本来、トップダウンに概念レベルから徐々に決定されるべきものといえる。つまり、概念レベルにおける実体(関

連)の発生, 死滅および変化に関する規則や制約は論理レベルにおけるデータの生成, 削除および更新に関する規則や制約と無縁である。概念レベルの一貫性制約は論理レベルにおける一貫性制約を支配し, 論理レベルの一貫性制約は物理レベルにおける一貫性制約を支配する。

操作部設計とは, 構造部と一貫性部の制約に基づき, その制約を満足するデータ操作を決定することである。データ操作は大きく, 1次データ処理と2次データ処理に分けられる。1次データ処理はデータの発生, 消滅, 更新を行う処理であり, その要件はデータを活用する利用者ニーズとは独立に, 実在界における実体(関連)の発生, 死滅, および様々な変化に従属させて決定されるべきものといえる。したがって, 1次データ処理の処理要素はデータ構造部と一貫性制約から一貫に導くと共に, データ構造部を構成するデータ型ごとにユニークに結合させる方式が望まれる。データ型とデータ操作を結合させる概念は[RISK75], [GUTT77]等によって抽象データ型として提案されている。

データモデルのレベル別にこれらの設計過程が存在する概念を図2に示す。

3. データ構造部設計

3.1 ERモデルによる構造最適化

概念レベルにおけるデータ構造部は実体とその関連の認識に基づくERモデルによって表現される。構造部設計ではERダイアグラムによる実体(関連)の表現だけでなく, 構造的最適化が行われる。正規化と抽象化はその基本的手法である。

(1) 正規化

正規化とは, 対象を認識するのに2つ以上の異なる事実が混在しないようにすることである。対象が正規化されない場合, 構造が冗長になるばかりでなく, データ操作の完全性も損われることが削除モデルにおいてよく知られている。

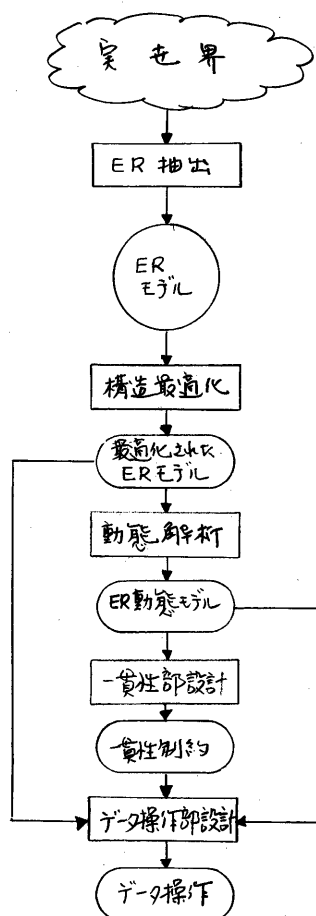


図1. データモデリング手順

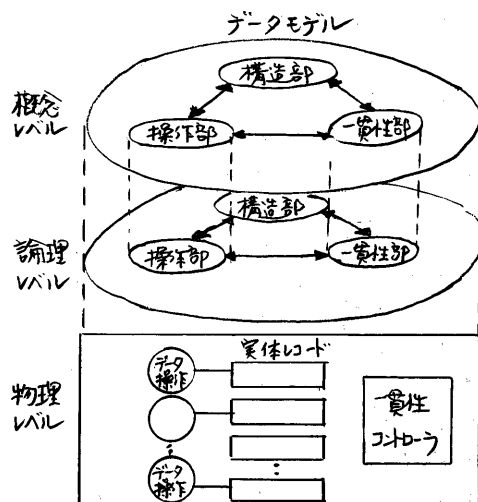


図2. データモデルレベル

関係モデルにおける正規化概念をERモデルに用いる。即ち実体集合E(関連集合R)においてE(R)の識別子に完全関数従属であるとき、E(R)は正規化されているという。E(R)における属性間の関数従属および完全関数従属の概念は、属性に対応する値集合の間の関数従属および完全関数従属として定義される。Rの識別子としては、Rによって関連づけられる実体集合の識別子と考える[SAKA80]。簡単な例を示す。

次のような実体(関連)の認識と関数従属性が与えられるとする。ここで、 $X \rightarrow A|B$ は属性A, Bが属性Xに完全関数従属であることを表わす。

実体集合

教育(科目#, 教師#, 学生#, 科目名, 要求事前修了科目, 単位, 教科書名, 著者, 教師名, 学生名, 成績)

関数従属性

科目# $\rightarrow$ 科目名 | 要求事前修了科目
| 単位

教師# $\rightarrow$ 教師名

学生# $\rightarrow$ 学生名

科目#, 教師# $\rightarrow$ 教科書名 | 著者

科目#, 学生# $\rightarrow$ 成績

教科書名 $\rightarrow$ 著者

実体「教育」を正規化すると次のようなスキーマが得られる。

実体集合

科目(科目#, 科目名, 事前要求修了科目#, 単位)

教師(教師#, 教師名)

学生(学生#, 学生名)

教科書(教科書#, 教科書名, 著者)

関連集合

講義(教師, 科目, 教科書)

履習(学生, 科目: 成績)

(2) 抽象化

抽象化とは、業務システムで扱うに必要十分な程度に抽象化された実体(関連)認識に対して、その構造的特性を明らかにすることである。抽象化レベルの階層化には汎化と専化、実現値化、および集約化の手法を用いる。

① 汎化と専化 (generalization - specialization)

汎化とは、あるカテゴリに1つ以上の類似の実体集合を集めて1つのクラスを作り、これを単一の实体集合とみなす抽象化をいう。逆に単一の实体集合とあるカテゴリに1つ以上の類似の実体集合に類別することを専化という。例えば、「科目」と「必須科目」、「選択科目」は互いにカテゴリ「科目種別」によって汎化と専化の関係にある。

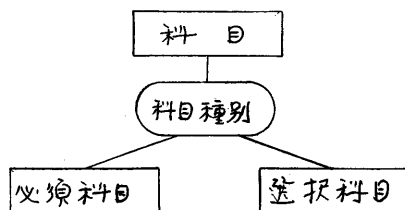


図3 汎化と専化

② 実現値化 (instantiation)

実現値化とは、ある抽象レベルの実体(関連)E1の型とみなし、その実現値と下位の実体(関連)として定義することである。この手法は多対多の対応で実体集合を関連づける関連集合と独立な実体集合に変換することと意味する。

実体集合EとFが関連集合Rによって多対多の対応で結ばれているとき、Eの実体E_iに対して、

$$\text{集合 } R/e_0 = \{(e_0, f) \mid f \in F, (e_0, f) \in R\}$$

を R にたいする e_0 の下別実現値集合とよぶ。ここで $(e_0, f) \in R$ の実現値を表わす実体と考える。 R を n のような実体を要素とする実体集合と考へ、 E の要素 e_0 に e_0 の下別実現値集合 $R/e_0 (\subset R)$ を対応させることを R に関する E の下別実現値化とよぶ。

③ 集約化 (Summalization)

実体集合がある基準に1だが、2, いくつかのグループに類別されるとき各グループに対してその集約値をもつ実体を定義することを E の集約化という。集約化は次の手順に1だが、2行われる。

- i) 実体集合 E を互に素な部分集合 $E_1, E_2, \dots, E_n$ に類別する。
- ii) 各 E_i に対して、その集約値をもつ実体 $\bar{e}_i$ と考へ、そこから構成される実体集合 $\bar{E} = \{\bar{e}_i\}$ を定義する。
- iii) $\bar{E}$ と E を関連集合 SUM-OF- E として関連づける。SUM-OF- E は $\bar{e}_i$ と e_i の全要素を関連づける。

図4は集約化の例を示すものである。[学期内科目]は[科目]を実現値化して得られた実体である。[科目記録]は同じ科目井によって類別された[学期内科目]の集約値をもつ実体である。[学期内科目]は当該学期内で管理対象として存在し、学期終了と共に消滅する一過の実体である。

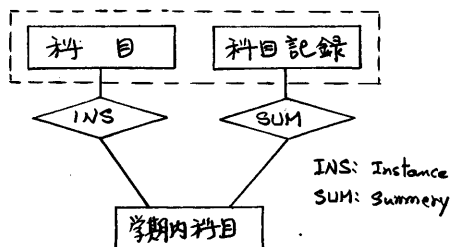


図4. 実現値化と集約化

3.2 動態モデリング

実体(関連)は時間と共に変化する。ERモデルは、その中で実体(関連)が発生し、変化し、消滅する枠組を表わすものである。

1つ。実体集合に属する個々の実体は、それぞれ固有の動態(behavior)をもつ。大学にたいして「学生」は入学し、と発生し、学期ごとに種々の授業に出席して単位を取得し、卒業と共に消滅する。このような動態は、ある要因によつて引起される状態(state)の推移と、その状態推移をもたらし作用によつてモデル化される。状態推移をもたらし作用をトランザクションとよぶ。

1つ。実体型に関する状態の集合とその状態推移をもたらしトランザクションの集合を動態とよぶ。このような動態の記述はデータ操作部や貫性部を設計するための有力な制約となる。動態の記述は、データ構造部の一部としてERスキーマに定義されるべきものと考えられる。1だが、2, ERスキーマは図5に示すように構造記述部と動態記述部から構成されるものとなる。

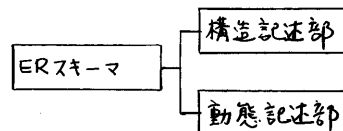


図5 ERスキーマの構成

動態、解析は、状態変化の主体である実体、記述と状態変化とトランザクションの関連を記述することが前提となる。その手順は次の通り、とされる。

- (1) 構造的に正規化され、かつ業務環境に近い抽象レベルをもつERモデルの設計
- (2) 実体(関連)集合の動態を調べ、ER動態図を作成する。

- (3) 動態で構成する状態の記述
- (4) トランザクションの正規化
- (5) トランザクションの記述

3.2.1 ER動態図

解析の対象となるシステムの動的性を記述する方法には[GRIN66], [RIDD78], [ROSE82]等が, イベント概念を用いた手法を示している。これらの手法はシステムの動態を, 主としてそのプロセスに着目して解析することを目的としている。しかし, データ中心アプローチではプロセスの解析に先立ち, データの解析を行う。したがって, データが反映している実体の動態を記述し, 解析する手法が重要となる。

データの構造的な特性を概念レベルで捉えるにはER図を用い, その上に各実体の動態を記述する手段としてペトリネット記法を用いる。ペトリネットによる動態記述をもつER図をER動態図(ER behavior diagram)とよぶことにする。

ペトリネットはP節点(円で表示)とT節点(矢線で表示)の2種類の節点をもつ有限有向2部グラフである。ここではP節点を状態, T節点をトランザクションを表わすものとして用いる。

図6はER動態図の概念を示すものである。

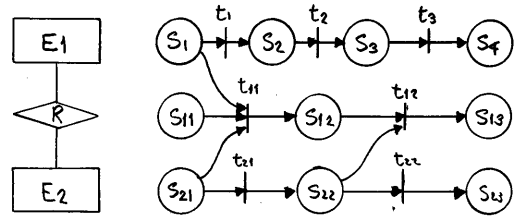


図6 ER動態図の概念

ER動態図では, 各実体(関連)に対応させたその実体(関連)がとり得る有意な状態をP節点として表わす。個々の実体(関連)に対応した一連の状態を実体生涯歴(entity life history)とよぶ。図6における実体[E1]の生涯歴は/S1/+S2/+S3/+S4/である。

状態の推移をもつトランザクションはT節点で表わす。t1, t2, t3はS1, S2, S3, S4の状態推移を引起すトランザクションである。矢線の左側から入る円が入力状態, 右側に出る円が出力状態となる。一般に, 1つの出力状態は1つの入力状態のみをもつとは限らない。複数の状態の同時成立を必要とすることもあり, 例えば, S12の入力状態はS1, S11, S21である。

トランザクションTの入力状態をX, 出力状態をYとすると, このトランザ

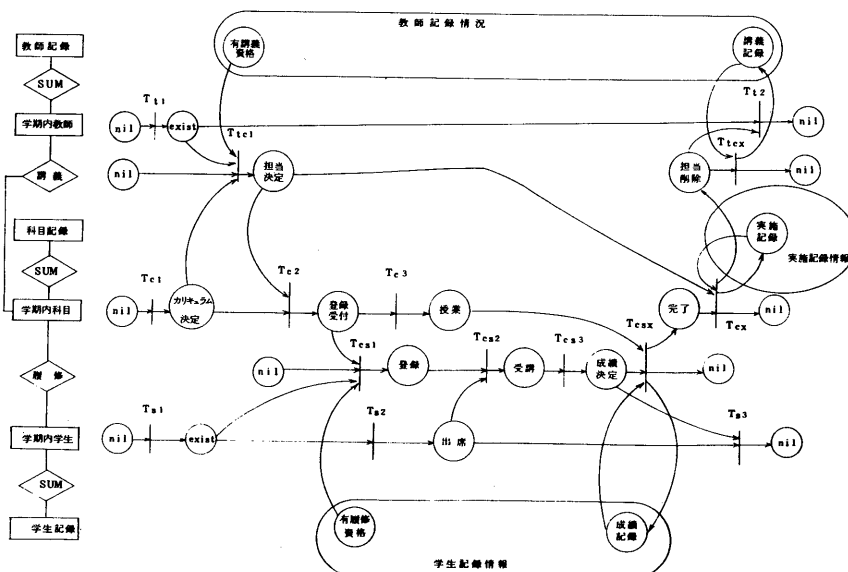


図7. ER動態図
《5》

クションは $T(X|Y)$ と書くことができる。図7は大学におけるER動態図の例である。

3.2.2 動態記述

ER動態図に示された実体(関連)の状態およびトランザクションについて正確な定義を行なうことで、動態の解析には不可欠となる。ER動態図における各状態とトランザクションに関する定義と動態記述とをよぶ。ER動態図と動態記述はERスキーマにおける動態記述部を構成する。

動態記述は大きく状態記述とトランザクション記述に分けられる。

(1) 状態記述

状態記述は実体(関連)の属性と値から構成される論理式によって表現される。

例 $\text{state}\langle\text{履修}\rangle = / \text{受講} / : \text{出席日数} \geq 20$

即ち、関連 $\langle\text{履修}\rangle$ の状態/受講/は論理式、出席日数 ≥ 20 によって定義される。論理式の項には次のような実体(関連)に関する述語が含まれる。

- ① 実体E(関連R)の存在を表わす状態の表現

$\text{state}[E] = / \text{exist} / : \text{exs}(E)$

$\text{state}\langle R \rangle = / \text{exist} / : \text{exs}(R)$

- ② 実体E(関連R)の非存在を表わす状態の表現

$\text{state}[E] = / \text{nil} / : \text{nil}(E) = \sim \text{exs}(E)$

$\text{state}\langle R \rangle = / \text{nil} / : \text{nil}(R) = \sim \text{exs}(R)$

- ③ 属性または属性集合が値をもつ(has value)という状態の表現

$\text{state}[E] = / S_i / : \text{hr}(v_1, v_2)$

$\text{state}\langle\text{履修}\rangle = / \text{登録} / : \text{hr}(\text{登録日付})$

- ④ 属性の値が変化した, $\text{hr}(\text{new Value})$ という状態の表現

$\text{state}[\text{学生記録}] = / \text{成績記録} / : \text{nv}(\text{履修済科目\#})$

ER動態図では同一実体(関連)の状態・集合を抽象化された1つの状態として示すことがある。たとえば $\langle\text{履修}\rangle$ の状態、 $/ \text{登録} /$ 、 $/ \text{受講} /$ 、 $/ \text{成績決定} /$ とよめ21つの状態、 $/ \text{履修状況} /$ と示すことがある。これを抽象化状態という。

(2) トランザクション記述

ある特定の実体(関連)に関する状態の推移をもたらす作用をトランザクションとよぶ。トランザクションの記述は入力状態と出力状態の対応に関する記述と個々の実体(関連)に対する操作の記述とから構成される。次の例はトランザクション T_{cs4} の入出力状態対応に関する記述である。

$T_{cs4}(X_{cs4} | Y_{cs4})$

$X_{cs4} = (\text{state}\langle\text{履修}\rangle = / \text{成績決定} /)$

$Y_{cs4} = (\text{state}\langle\text{履修}\rangle = / \text{nil} /)$

トランザクションの操作に着目した記述は単純トランザクション記述の列として表現される。単純トランザクション(S)は1次形式または2次形式のいずれかの形式をもつ。

① 1次形式

$S : \text{operator } U \text{ または } \text{operator}(U, \text{state})$

operatorは操作、Uは操作の対象となる実体(関連)集合あるいは部分集合である。operatorは, get, create, delete, modifyのいずれかである。stateは操作の結果生成されるUの状態である。1次形式のoperatorの意味は次の通りである。

(a) $S : \text{get } U$

Uの属性に関する条件を満たすUの部分集合を取り出す。

(b) $S : \text{create}(U, \text{state})$

実体(関連)を新たに創生しその状態をstateにして、Uに追加する。

(a) または (b) の形式によつて U から取り出される、または U に追加される U の部分集合を単純トランザクションの目的空間といい、 S_i で表わす。

(c) $S : \text{delete } S' \text{ または } \text{modify}(S', \text{state})$

1 次または 2 次形式の単純トランザクション S' の目的空間 S' を消去または変更する。delete の場合 S' の状態は /nil/ となり、modify の場合は S' は state で示される状態になる。

② 2 次形式

$S : \text{operator } U \text{ through } R(S_1, S_2, \dots, S_k)$

または

$\text{operator}(U, \text{state}) \text{ through } R(S_1, S_2, \dots, S_k)$

operator は get, create のいずれかである。 U は操作の対象の实体 (関連) 集合, state は操作の結果発生する U の状態を表わす。 R は参照される関連集合, $S_i (i=1, 2, \dots, k)$ は S 以前に実行される単純トランザクション S_i の目的空間である。2 次形式は次の意味を表わす。

(a) $S : \text{get } U \text{ through } R(S_1, S_2, \dots, S_k)$

$S_1, S_2, \dots, S_k$ と R によつて関係づけられた实体集合 U , または $S_1, S_2, \dots, S_k$ を含む関連集合 $U = R$ を対象として, U の属性に関する予えられた条件を満たす U の要素または部分集合を取り出すことを意味する。

(b) $S : \text{create}(R, \text{state}) \text{ through } R(S_1, S_2, \dots, S_k)$

$S_1, S_2, \dots, S_k$ を結合した関連集合 R の要素または部分集合を創成し, その状態を state にし, R に追加することを意味する。

次の例は単純トランザクション記述

による関連集合〈履修〉に属するトランザクション T_{cs1} を記述するものがある。

$SC1 : \text{get} [\text{学期内科目}]$

$SS1 : \text{get} [\text{学期内学生}]$

$Ssrl : \text{get} [\text{学生記録}] \text{ through } (Ss1)$

$Scs1 : \text{create} (\langle \text{履修} \rangle, / \text{登録} /) \text{ through } \langle \text{履修} \rangle (SC1, SS1)$

$T_{cs1} : SC1 ; SS1 ; Ssrl ;$

if state ($SC1 = / \text{登録受付} /$)

^ state ($Ssrl = / \text{有履修資格} /$)

then $Scs1$

3.3 動態の解析とトランザクション正規化

ER 動態図に示された实体 (関連) の動態は状態とトランザクションの集合として記述される。正しくデータモデルにおける一貫性部やデータ操作部を設計するには, データ構造部が正規化手法等によつて最適化されたように, 動態によつても最適化が必要となる。トランザクション正規化はそのための手法である。

トランザクション正規化とは, トランザクションを構成する基本的要素を発見する過程である。つまり, トランザクション $T(X|Y)$ に於いて Y がただ一つの实体 (関連) 集合に関する ER 状態のみから成るとき, T は正規化されるという。このことは T の作用対象がただ一つの实体 (関連) 集合に含まれることを意味する。

もし, トランザクション $T(X|Y)$ が正規化されないならば, Y は n 個の異なる实体 (関連) 集合に関する状態 $Y_1, Y_2, \dots, Y_n$ に分けられる。このとき T は n 個の正規化されたトランザクション $T_1(X_1|Y_1), T_2(X_2|Y_2), \dots, T_n(X_n|Y_n)$ に分解されることがある。

元は出力状態に発生させるに必要な極小なER状態から成る。例として図のトランザクション $Tcsx(Xcsx|Ycsx)$ の出力状態 $Ycsx$ には「学期内科目」、「学生記録」、〈履修〉に関するER状態が含まれている。このようなとき $Tcsx$ を次のような3つの正規形トランザクション $Tc4, Tcs4, Tsr1$ に分解することができる。

(a) $Tc4(Xc4|Yc4)$
 $Xc4 = (\text{state}[\text{学期内科目}] = \text{授業}/)$
 $\wedge (\text{state}[\text{履修}] = \text{成績決定}/)$
 $Yc4 = (\text{state}[\text{学期内科目}] = \text{完了}/)$

(b) $Tcs4(Xcs4|Ycs4)$
 $Xcs4 = (\text{state}[\text{履修}] = \text{成績決定}/)$
 $Ycs4 = (\text{state}[\text{履修}] = \text{nil}/)$

(c) $Tsr1(Xsr1|Ysr1)$
 $Xsr1 = (\text{state}[\text{履修}] = \text{成績決定}/)$
 $\wedge (\text{state}[\text{学生記録}] = \text{成績記録}/)$
 $Ysr1 = (\text{state}[\text{学生記録}] = \text{成績記録}/)$

図8は図7に示けるトランザクションを正規化した結果を示すものである。正規化前後のトランザクションの対応は次のようになる。

正規化前	正規化後
$Ttcx$	$\rightarrow Ttc3, Ttr1$
Tcx	$\rightarrow Tcs, Tcs2, Tsr1$
$Tcsx$	$\rightarrow Tcs4, Tc4, Tsr1$

先に述べたトランザクション記述はこのような正規化を受けけるトランザクションに分解される。

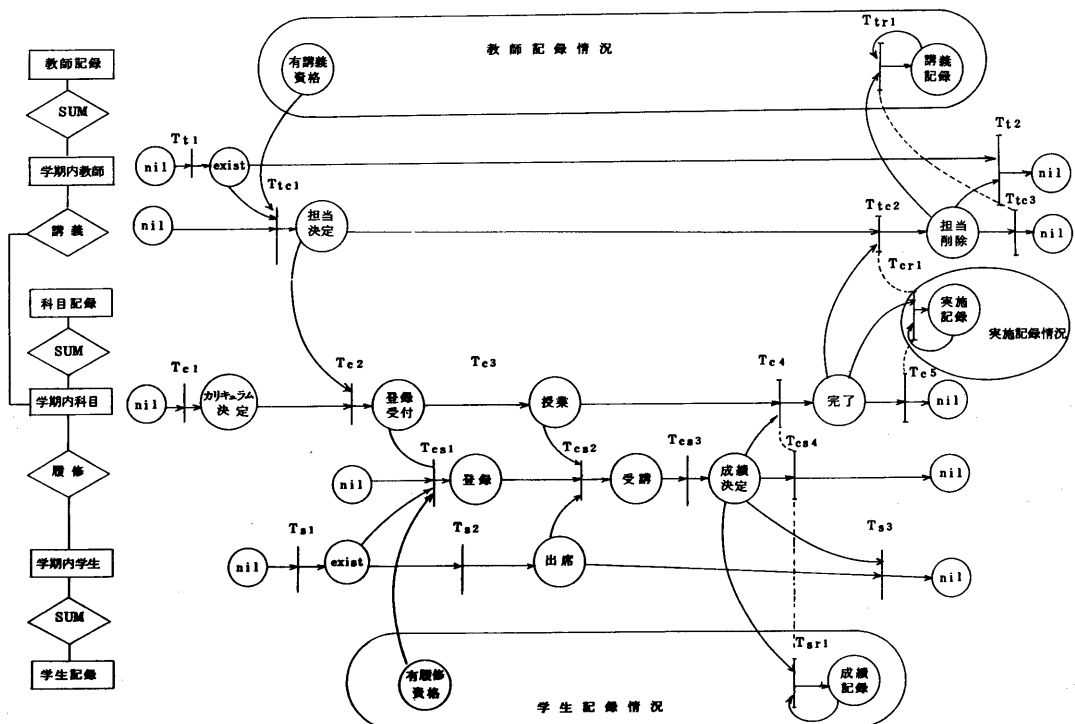


図8. 正規化したER状態図
 (8)

4. 動態モデルに基づく一貫性とデータ操作部の設計

4.1 データ中心システム

実体(関連)の記述とその動態に関する記述から成るデータ構造部(ERスキーマ)が設計された後、このことからデータ一貫性部と操作部の設計を行うことが可能となる。最適化されたデータ構造部に基づいてデータ操作部等を設計するためには、システム全体の構築に関する次のような前提が必要となる。

- (1) データ操作を1次データ処理のためのものと2次データ処理のためのものとに分離する。
- (2) 最適化されたデータ構造部を中心に1次データ処理を先に設計し、1次データ処理の一貫性を確保した後、2次データ処理を設計する。
- (3) 1次データ処理はデータ構造部に定義されているデータ型に基づいて1つに統合されるように設計され、データ型とデータ処理の統合を図る。

このような方式をもつシステムをデータ中心システムとよぶことにする。

データ構造部に関するERスキーマには実体(関連)を反映したデータ型が定義されている。このようなデータ型を実体レコード型(entity record type)とよぶことにする。

1次データ処理とは実体レコード型に基づいて、その型に沿ったレコードの生成、更新、削除を制御するデータ操作の集合をいう。2次データ処理とは実体レコード型に基づいて、データ問合せ処理を行い、関係演算として形式化が行われる操作の集合となる。

図9はデータ中心システムの基本構成を示すものである。P₁, P₂, ..., P_nは同一実体レコード型を対象とするデータ処理で正規化されたトランザクションごとに用意されるものである。P₁, P₂, ...,

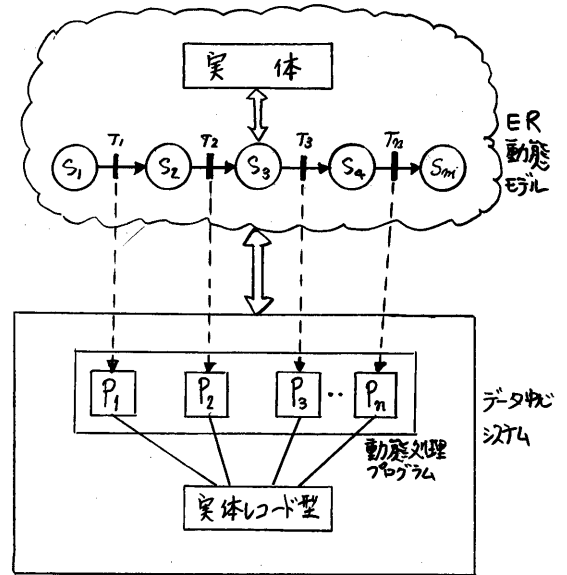


図9 データ中心システム

P_nをトランザクション処理モジュールとよぶ。トランザクション処理モジュールは対象とする実体レコード型ごとに集められ一つのプログラムとして実現される。このようなプログラムをER動態処理プログラムとよぶ。データ中心システムはこのような動態処理プログラムと実体レコードの集合、さらにER動態処理プログラムを制御するER動態スーパーバイザと2次データ処理のための問合せ処理プロセスから構成される(図10参照)。

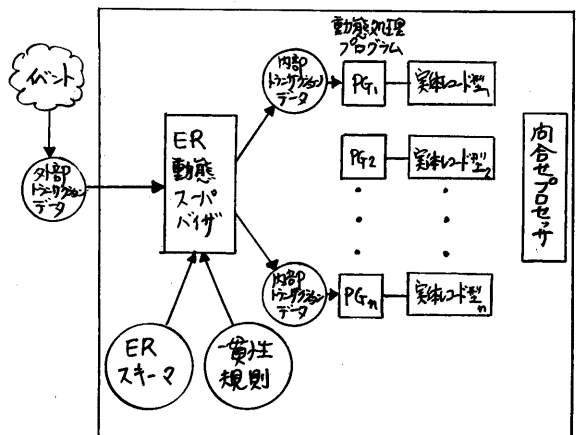


図10 データ中心システムの構成

4.2 ER動態スーパーバイザと一貫性部の設計

ER動態スーパーバイザはトランザクションの実行順序や1つのトランザクションの発生による波及的に起動されるトランザクションの監視、制御を統括する機能である。ER動態スーパーバイザは実世界における事象の発生を外部トランザクションデータの投入によって察知し、実体レコード型ジヒに内部トランザクションデータの生成と動態処理プログラムの実行制御を行う。ER動態スーパーバイザはシステムに内蔵された1次データ処理の集中制御を行う一貫性制御機能となる。したがってER動態スーパーバイザにはER動態記述と1次データ処理の一貫性制約記述を与えられなければならない。ER動態記述はトランザクションの同期、依存関係を判断するための情報としてER動態スーパーバイザに与えられる。1次データ処理の一貫性制約記述はトランザクションデータの処理の妥当性や正当性を検証するための規則や命題として与えられる。その規則、命題はER動態記述を入力情報とする設計作業による定義されるものである。規則、命題の形式化については不明な点があり、実体レコード型ジヒにその実体レコード型に対応する正規形トランザクション別に、制約を属性と属性値に関する命題として定義すべきであることは明確といえる。何故ならばトランザクション別、あるいはプロセス別の制約定義は、規則や命題自体の一貫性を損いやういふことがある。ER動態モデルをもつことは、実体(関連)

[参考文献]

1. [NOLAN79] R.L. Nolan, "Managing the crises....", Harvard Business Review, 3-4 (1979)
2. [MART82] J. Martin, "An Overview Plan", Computerworld Oct. 4, (1982)
3. [FINK81] C. Finkelstein, "Information Eng.", Computerworld, 5/14/81 (1982)
4. [JACK76] M.A. Jackson, 構造化設計法..., 日知堂出版 (1976)
5. [WARN80] J.D. Warren, プログラム設計法, 日知堂出版 (1980)
6. [MYER75] G.J. Myers, Reliable Software..., Prentice-Hall (1976)
7. [CODD82] E.F. Codd, "Relational Database...", CACM, Vol. 25, No. 2 (1982)
8. [RISK75] B. Liskov, "Specification tech...", IEEE Trans. on Soft. Eng. 3 (1975)
9. [GUTT72] J.V. Guttag, "Abstract data...", CACM, 7 (1972)
10. [ESAK80] H. Sakai, "Entity Relationship App...", Proc. ACM SIGMOD (1980)

ジヒに一貫性規則、命題を検討するための有力な手段となる。

4.3 データ操作部の設計

ER動態スーパーバイザはトランザクション実行制御と一貫性検証を統括して行うならば、システムでプログラム設計作業の大勢を占める。ER動態スーパーバイザの一貫性制御論理の設計は不要となる。

1次データ処理の設計はERスキーマに示された実体レコード型に対応するトランザクションジヒのトランザクション処理モジュールを設計するものとなる。トランザクション処理モジュールは、内蔵の入力データエントランザクションデータとして、内蔵の実体レコードを更新するものとなることから、その設計作業は極めて容易になると共に、設計者の主観に左右されない一貫性の高いものとなる。

5. おわりに

動態記述をもつERスキーマとER動態スーパーバイザはシステム構造の単純化と信頼性向上に有効であるだけでなく、データ中心アプローチを実現するためにも重要な手段となる。今後、ER動態モデルから一貫性制約を抽出し記述する方法に関する研究が期待される。