論文

材料データシステムの統合化(2) メタデータの表現と管理[†1]

芦野 俊宏[†2] 岩田 修一[†3]

材料データベースは，所在が分散しており，使用目的も多岐に渡るため，データ構造，運用形態などが様々である．このような複数の異種材料データベースを統合利用・運用するための方法論を提示することを目的として，クライアント/サーバ型のシステムを設計し，データ型の拡張，メタデータ管理，ユーザインターフェイスについての要件を定義し，プロトタイプシステムの開発を行なった．

前報 "材料データシステムの統合化(1)" では，異種の材料データベースを統合し，数値シミュレーションを関係型データベースの枠組で扱うためのシステムのプロトタイプについて述べた．本論文では，材料データシステムにおけるメタデータ管理について述べる．

異種データベースを統合利用するためには，各々のデータベースに収められているデータを記述するメタデータをシステム内で管理し，これを元に必要なデータの変換を行なう必要がある．これを実現するために複数のデータベースサーバ(DS)とクライアントとの間にメタデータサーバ(MS)を置き，メタデータを集中管理し，これを用いてデータの変換を行なう手法を開発した．

1 はじめに

工学分野においては基礎データとして材料データの重要性が高い．このため，材料データを計算機可読化して数値データベースとして扱おうとする試みが数多く行なわれている．しかし，これらは各々の研究機関，企業などが独自に，内部での利用を目的として行なっている場合が多いため，データ項目名，データ構造，単位など独自の形式でデータベース化されている．

また，独立に開発されているために基本的なデータに関しては重複してデータベース化されている部分も多い．重複を排除し，データベースを有効利用するためには，データの所在・種類・内容などを記述したメタデータを整備していくことが必要であり，このためにCODATA, NIST, ECなどによってデータベースディレクトリが作成されている．[1,2,3]

このようなハードコピーの形で与えられたメタデータ情報に対し，メタデータを計算機可読な形で整備することによって，近年発達の著しい広域計算機ネットワークを用いた材料データベースの利用をより効率的に行なうことが可能になる．

しかし，各々のデータベースへのアクセス手順の違い，データ項目名の与え方のルールの違いなどにより，単にゲイトウェイの機能を与えるだけでは不十分であるため，ログイン手順の統一やデータ項目名の自動変換，さらには数値データの表現されている単位の自動変換など，より高度なシステムの構築が望まれる．このような方向を目指したプロジェクトとしては，

・MIST/NBS, DOE(U. S. A.)-NMPD Network, Inc.
・DEMO/CEC D. G. XIII(EC)

等がある．[4,5,6]

前報 "材料データシステムの統合化(1)" [7]では，異なった関係型データベース管理システム(RDBMS)に対して仮想的なSQLサーバのインターフェイスを与えることによって異種データベー

---

†1 Integration of Materials Data System. (2) Representation and Management of Metadata.

†2 Toshihiro Ashino, 新日本製鉄(株)先端技術研究所, Advanced Materials & Technology Research Labs, Nippon Steel Corporation

†3 Shuuichi Iwata, 東京大学人工物工学研究センター, Research into Artifacts, Center for Engineering

スのアクセス手順を統一化するためのシステム設計について述べた.

本論文では,データ項目の名前,表現されている単位,データ構造などを自動的に変換し,異なった構造をもつ材料データベースに対する透過的なアクセスを可能とするシステム開発のを目的として,材料データに関するメタデータの表現を与え,さらにこれを用いてデータの変換を行なうシステムの設計を行なう.

また,シリーズ論文の(3)として,実装例による検索とシステム評価,本格的な材料データシステムの展望について述べる予定である.

2 材料データシステムにおけるメタデータ

2.1 メタデータの分類

材料データにおけるメタデータは以下の三つのレベルに分類することが出来る.

1) データの項目名,データ構造,所在などデータを検索するレベルでの表現.
2) 数値データの単位系,表記法など数値データの内部表現に関するもの.
3) 信頼性など入力するデータの評価に関するもの.

これらは個々のデータの内容に関する記述であり,データ構造とは独立に管理されるものである.これに対して,関係型データモデルでは,データ項目間にある "is a", "a kind of" といったような意味的な相互関係をデータ構造によって陽に表現することができないために,上記以外に

4) データ項目間の意味的な相互関係

をメタデータの一種として考える必要がある.

データ項目間の関係をデータとは別に管理するフロントエンドを作成し,データ構造・問合せを相互変換することによって,広く普及している関係型データベースを,実体-関連 (Entity-Relationship) モデルなどより記述力の高いデータモデルとして扱うことができる[8].本研究では,4) に関しては3.2.3 項に示したような単純な階層構造を扱うにとどめた.

2.2 メタデータを用いたデータベース統合化

前節において述べたデータの内容に関する 3 種のメタデータのうち,1) と 2) に関しては,適切な辞書を作成し,メタデータに従って変換を行なうことによって異なったデータベースに対し透過的なアクセスを実現することが可能である.

これに対して,3) のデータ評価に関しては,計算機上で扱うことの非常に困難な問題である.これには,データの入力段階での評価基準の標準化を行ない,データの評価によってデータベース自体を多層化する必要があると考えられる.

本論文では,1) と 2) の二つのレベルのメタデータの管理とそれに基づいたデータ変換について述べる.ただし,データ構造の変換に関しては,関係型データモデルに限っている.

データ変換を行なうためには,ネットワーク上でのデータ項目を一意に識別する必要がある.このためには,

1) そのデータの存在する計算機のホスト名
2) 当該ホスト上に複数の DBMS が稼働している場合はその識別名
3) 一つの DBMS 内で用いられるデータベースの識別名
4) データベース中の表の名前
5) 表の中でのデータ項目名

という 5 つのメタデータを相互接続する全てのデータベースの項目について整備することが必要となる.また,各々のデータ項目の属性としては,そのデータ型と,数値データの場合には用いられている単位を必要とする.

また,関係型 DBMS のデータ構造を変換するためには,データベースの複数の表の間の関係を記述する必要がある.関係型データモデルにおいては,材料データのような複雑な構造をもつデータは一般に図 1 に示したように複数の表に分割して扱われる.[9]このような表の分割の方法は各々のデータベースによって異なる.

複数の表にまたがって属性を関係づけ,検索を実行する場合には,キーとなる属性を用いて複数の表を結合する.異種データベースを統合利用するためには,この結合の操作を自動的に行なうことが必要であり,そのためにキーとなる項目とそ

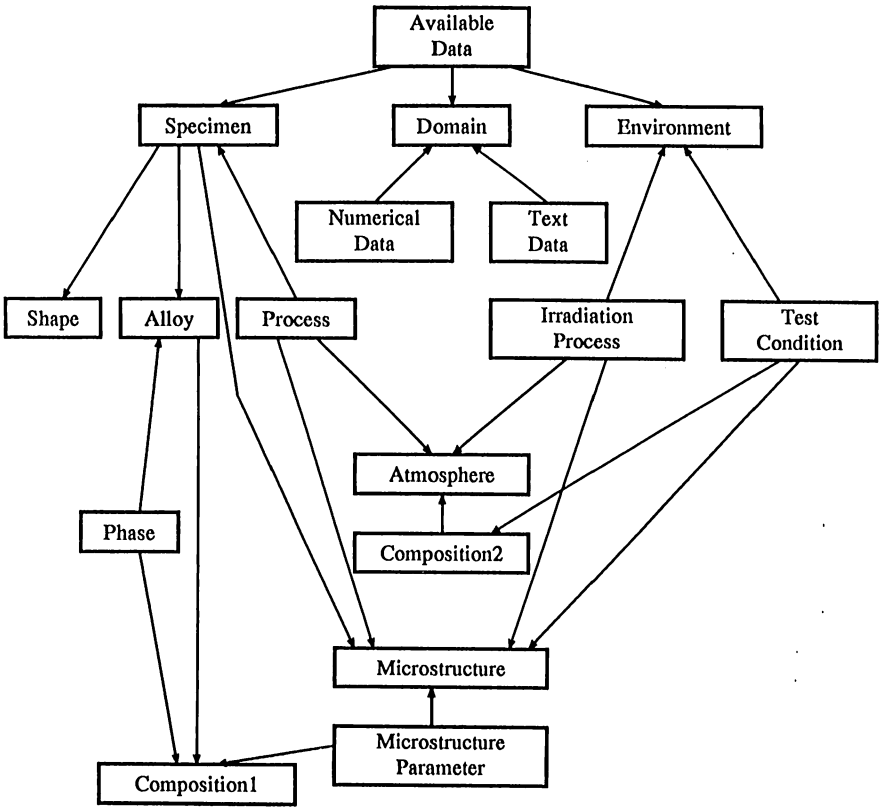

図1 放射線照射実験データベースのデータ構造[9]

れによって結合することの出来る表との関係を記述する必要がある.

3 メタデータ管理の実装

文献7)において述べたように, システムは図2のような構成になっている. Database Server(DS)は異なったRDBMS各々について実装され, 仮想的なSQLサーバの機能を提供する. Metadata Server (MS)は複数のDSとクライアントの間に介在し, メタデータに基づいてクライアントからのデータ検索要求を各々のDSで解釈される形式に変換して送信する. メタデータの管理とこれを用いたデータ変換は, 拡張データ型と共にMSに実装されている. また, 図2には示されていないが, ユーザのアクセス制御を行い, これらのサーバプロセスの起動を行うためのCommunication Server(CS)が存在している.

3.1 データ表現

MS内部ではメタデータを図3に示すstructure, directory, dependencyという三つの表を用いて管理している.

3.1.1 structure - 材料データの意味

structureは, データ項目に対応する物理的特性など, 材料データ中での意味を記述するための表であり, データ項目とそれに対応する概念を明示的に管理することを目的としている. structureのフィールドであるlabel, id, attr, type, unitについて以下に述べる.

labelは, "Yield Strength", "Material Name"など, データの内容を分かり易く表記した文字列であり, 一般にデータ項目名に用いられる略語とは異なっている. labelとidは一意に結び付けられ

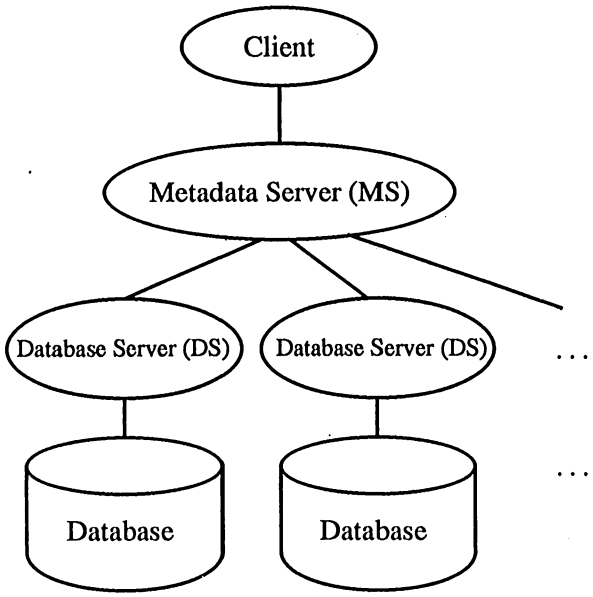

図2 プロトタイプシステムのサーバプロセス構成

structure

label	*id*	*attr*	*type*	*unit*
Material Name	1	0	string	–
Yield Strength	3	0	float	MPa

directory

host	*dbms*	*dbname*	*tname*	*cname*	*type*	*width*	*unit*	*id*
zodiac	oracle	generic	material	mname	string	30	–	1
zodiac	oracle	generic	material	YS	number	–	N/mm^2	3
libra	ingres	stainless	property	yield_strength	float	–	ksi	3

dependency

key1	*key2*	*table1*	*table2*
materialid	materialid	material	property

図3 メタデータ管理のために用いられる3種類の表

ており，structure のキーとなっている．また，id は structure と directory とを結び付けるためにも用いられる．現在は MS 一つにつき，一度に扱える structure は一つ(起動時に切り換えることは可能)なので，id と label との対応は一組だが，複数の表を管理するようにシステムを拡張することによっ

map

table identifier				*id list*			
host	*dbms*	*dbname*	*tname*	1	2	3	.....
zodiac	oracle	generic	material	mname	figno	YS	.....
libra	ingres	stainless	fig1	name	0	0	.....
libra	ingres	stainless	property	name	0	yield_strength	.....

図4 メタデータより生成されるデータマップ

て，同じ特性に対して別の呼び方がある場合，すなわち同義語の問題を扱うことができると考えられる．

また，各々の label に対して一般的に用いられるデータ型(type)と単位(unit)の情報があり，検索時にクライアントから変換の指定がなかった場合にデフォルト値として用いられる．

attr は，3.2.3 節において述べるデータの階層構造を生成するための属性で，材料データの意味を考えた場合に，「その特性の上位の概念」であると考えることの出来る structure 中のエントリの id 番号が入る．これにより，材料データの意味関係を階層的に記述するためである．本来，材料データの間にみられるような複雑な相互関係を表現するには，単純な階層構造のみの記述では不十分である．しかし，第一段階としてはシステム開発に重点を置き，データ項目の関係は階層構造に限定して扱うこととした．

3.1.2 directory - データの所在

directory は，データ項目の名前，所在，データ型，単位などのネットワーク上で特定のデータ項目を記述するためのメタデータを記述しており，9 つのフィールドを持つ．この中には，2.1 節で述べたデータ項目ををを一意に識別するための 5 つのメタデータ，すなわち，ホスト名(host)，DBMS 識別名(dbms)，データベース識別名(dbname)，表の名前(tname)，データ項目名(cname)が記述される．また，DBMS 上での表現に用いられているデータ型(type)，文字列など可変長のデータの場合にはその最大幅(width)という，データ型の変換に必要な情報を記述する．

ここでは id は一意には定まらない．id は structure で記述された各々の物理的特性の名前などについて一意に定められるからである．即ち，図 5 の例にあるように同じ Yield Strength という特性が複数のデータベースで YS, yield_strength のように別の略語を用いて記述されている場合，directory には各々について別のエントリがあるが，同じ id である 3 がつけられ，structure 上での YieldStrength という名称と結び付けることができる．

また，単位については，各々のデータベースで用いられている単位が文字列として記述されており，MS 内に別に用意された単位換算表を用いて structure で指定された単位に変換される．これによって，用いている単位系の違いを吸収することができる．

DBMS 識別名は，データ項目の識別に加えて，起動すべき DS を決定するためにも用いられる．DS は異なる DBMS に対して仮想化した同一の SQL サーバとしてのインターフェイスを与えるために，DBMS の種類毎に各々のライブラリを用いた異なるプログラムが必要である．いている．DBMS 識別名と DS の実行形式ファイル名との対照表は CS において管理されており，directory から検索要求対象となるデータ項目に対応する DBMS 識別名を与えることによって適切な DS を起動し，接続が確立される．

このようにデータの意味を表す structure とデータの位置を表す directory とを分割して表現することにより，各々のデータ項目間の同値関係をより明確にし，メタデータを材料データの意味から分離して管理することが可能となっている．

3.1.3 dependency - 表の間の関係

dependency は二つの表の間の依存関係を記述する．ここでは二つの表の名前と，それらを結合する

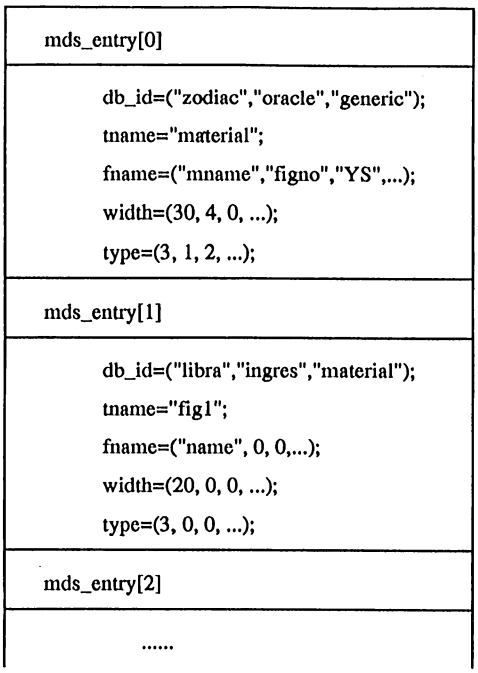

図5 データマップのMSにおける内部表現

ために用いられるキーとなる属性の名前が与えられる. 例えば, 図3の例に示したデータは, 二つの表 material と property が, where material. materialid =property. materialid という SQL 文を用いて結合することが出来ことを意味している. MS では, クライアントから複数のデータ項目の検索を要求された場合, directory の情報から同じ表に全ての項目がないことが明らかになった場合には dependency を参照し, 結合することの出来る表に該当する項目を探し, 全て見つかったところで各々の表を結合したビューを定義してそこで検索した結果を返す.

3.2 MS 内部におけるメタデータの取り扱い

3.2.1 データマップ

MS はメタデータの変換を効率的に行なうために, 起動時に表 structure と directory の情報から図4に示したような map と呼ぶ表を生成する. map は MS のメモリ空間内に生成されるため, 高速にメタデータの検索を行なうことが出来る. map は, 二つの部分に分けることが出来る.

一方は図4に示した table identifier の部分であり, もう一方は id list の部分である. tableidentifier はホスト名, DBMS 識別名, データベース名, 表の名前によって, 全てのデータベースの中の表を一意に指定する部分である. id list の部分の数は可変であり, structure で記述された全ての id 番号に対して, table identifier で指定された表の中にあるデータ項目名がどの id 番号に対応するかが記述されている. 対応するデータ項目がない場合には, ゼロを値として持っている. 例えば, 図3に示したデータからは, 図4のような map が生成される.

3.2.2 データマップの内部表現

図4に示した map は, MS 内部では図5に示したようにデータ構造体 mds_entry の配列として表現される. 配列 mds_entry の各々の要素には, 一つの表について, そこに含まれる項目名, データ型, 幅, 単位の情報が id をインデックスとした配列として格納される. クライアントからの検索要求は

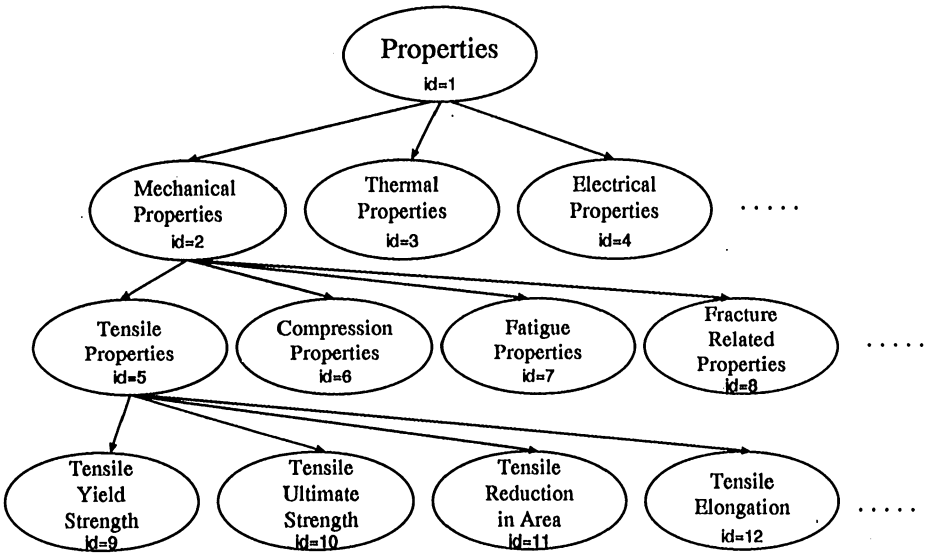

図6 材料の特性を階層構造としてまとめた例

structure の中での id の形で行なわれるため，これを現実のデータベースにおける項目名，データ型などに変換する必要がある．この変換は，map に含まれる全ての mds_entry 構造体について id をインデックスとして fname, width といった配列にアクセスすることで可能であり，パターンマッチを最小限に押さえることによってデータ検索のたびに行われるメタデータ検索の高速化をはかっている．

埋め込み型 SQL によるデータの検索は，検索する対象にカーソルを設定し，これを一行ずつ進めてデータを読み込んでいく(フェッチする)形をとるが，MS はこのカーソル設定の要求を受けた時点で，map を参照して対象となる表を探し，必要なら dependency に基づくビューの設定，id から実際のフィールド名への変換，DS への接続を確立する作業を行なう．カーソルをフェッチする要求は一行分ずつクライアントから MS へ送られる．複数のデータベースにまたがる検索の場合には map の中に書かれている順番に従って，各々の表の検索を順に行なう．

複数の DS に対してデータの検索を行う場合には，この操作は MS に検索結果のデータをキャッシュし，システムをマルチスレッド化することによって並列化し，高速に行なうことが可能であるが，現在の実装では行なっていない．また，map を MS のメモリ中に展開したために，データベースを変更した場合実際のデータ構造との矛盾が発生する可能性がある．MS を介しての変更であれば自動的にメタデータを更新することも可能であるが，一つのデータベースが複数の MS を介してアクセスされる場合や，MS を介在しない変更に関しては変更自体を検知することが困難であり，表の変更に対してメタデータを自動的に更新するにはトリガなどを用いてデータベースから DS, MS へ変更を知らせるメカニズムが必要である．

3.2.3 データ階層の実現

材料データベースのデータ項目について，図6に示したような包括的な概念から個々の項目についての階層構造を作っておくことによって検索対象を絞り込んでゆくために有用であると考えられる．このような階層構造を表すために MS 内において structure から生成される表が hierarchy である．本来，このようなデータ項目群の関係は網構造をもつと考えられるが，ここでは，取り扱いの単純化のためにただ一つの根をもった単純な木構造を仮定する．

map と同様，hierarchy は MS の起動時に struc-

hierarchy

label	*id*	*upper_class*	*lower_class1*	*lower_class2*	.....
Properties	1	0	2	3	.....
Mechanical Properties	2	1	5	6	.....
Thermal Properties	3	1	–	–	.....
Electrical Properties	4	1	–	–	.....
Tensile Properties	5	2	9	10	.....
Compression Properties	6	2	–	–	.....
Fatigue Properties	7	2	–	–	.....
Fracture Related Properties	8	2	–	–	.....
Tensile Yield Strength	9	5	–	–	.....
Tensile Ultimate Strength	10	5	–	–	.....
Tensile Reduction in Area	11	5	–	–	.....
Tensile Elongation	12	5	–	–	.....
...	...	...	...	...	.....

(a)

label	*id*	*upper_class*	*lower_class1*	*lower_class2*	.....
zodiac	1	0	8	21	.....
oracle	8	1	5	7	.....
...	...	...	...	...	.....

(b)

label	*id*	*upper_class*	*lower_class1*	*lower_class2*	.....
STEEL	34	3	18	48	.....
STAINLESS STEEL	18	34	9	10	.....
...	...	...	...	...	.....

(c)

図7 データ階層を表現するためにサーバ内部で生成される表 hierarchy. (a)特性の種類, (b)材料の種類, (c)データの所在による階層の表現.

ture をもとにメモリ空間内に生成される. structure の attr フィールドには上位の概念と考えられる id のみが書かれている. 図6で Properties に相当する最上位の概念については, attr の値がゼロになっている.

MS 内部での hierarchy は図7(a)のようなリスト構造として実装されており, 下位の概念(lower_class)の要素数は可変になっている. 単純な木構造を仮定しているため, attr の値をもとにリストを作るのは容易である. 木の根にあたる最も上位の概念に相当する項目はただ一つだけ決まり, そこでは upper_class の値がゼロになる.

図7(a)には, 図6の構造を表す hierarchy のデータを示す.

3.2.4 データ階層内の移動

クライアントがこの階層構造を扱うために, id を指定して階層構造の中を移動し, 現在の階層内に存在するデータの id とその label を取得するために表1に示すようなプロトコルが用意されている. これは, 木構造のファイルシステムを持つオペレーティングシステムにおけるカレントディレクトリの変更のためのシステムコールに類似している.

MDB_GET_LABEL は現在の位置とは無関係に, 指定された id に対応するラベルを返す. MDB_FIRST_LABEL, MDB_NEXT_LABEL, MDB_CHANGE_LEVEL は階層構造の中の移動と,

命令	動作
MDB_GET_LABEL	指定したid番号に対するラベルを返す
MDB_FIRST_LABEL	現在の階層における最初のラベルとそのid番号を返す
MDB_NEXT_LABEL	現在の階層における次のラベルとそのid番号を返す
MDB_CHANGE_LEVEL	指定したid番号のある階層へ移る
MDB_GET_FIELD	指定したid番号をもつ項目と同じ表にある項目のid番号を返す

表1 データ階層を辿るためのプロトコル

ラベルおよびidを調べるための命令である.

例えば, 図6の例において, 現在の位置が上から3番目のTensile Propertiesに始まる階層である場合, MDB_FIRST_LABELはTensile Propertiesというラベルと5というidを返す. この後続いてMDB_NEXT_LABELを実行すると順にCompressionProperties, Fatigue Propertiesを返す. MDB_CHANGE_LEVELはidを引数として取り, 階層を移動する. 指定したidが下位の階層を持つ場合にはその階層へ移動し, 持たない場合には指定したidを含む階層へと移動する. -1を指定すると一つ上の層へ移動する. 図6の例では, 5を指定するとTensilePropertiesの下位の階層であるTensile YieldStrengthを含む階層へ移動し, この直後に発行されたMDB_FIRST_LABEL命令は, Tensile YieldStrengthというラベルと9というidを返す.

また, MDB_GET_FIELDは木構造をデータベース中表によって与えられる関係をもとに横断的に検索するために用いられる. idを指定すると, マップを調べ, 指定したidをフィールドとして持つ表のその他のフィールドのidを順に返す. これにより, あるidに対して関連する項目としてどのようなものがあるかを取得することがで可能で, 指定したidのデータ項目に関連する異なった階層のデータを探し出すことができる.

3.2.5 利用目的に合わせたアクセスパスの提供

図6に示した例は材料の特性を機械的特性, 電気的特性などに分類していって最終的には特性の名前に至る階層構造である. しかし, システム管理者は図7(b)のホスト名→DBMS名→データベース名→表のようにデータの所在の情報による階層化を必要とするであろうし, ある応用に対して候補となる材料を絞るために利用する場合には, 図7(c)のような材料の種類による階層化が便利であろう.

このような場合のために, MSはstructureを切替える機能を持つ. デフォルトではstructureという名前の表を利用してmapやhierarchyを生成するが, 起動後, あるいは起動時にクライアント側から別の名前の表を用いてmapやhierarchyを生成するよう指定することが可能である. これによってフラットなデータベースに各々の目的に沿ったデータ階層構造を実際のデータ構造とは独立に定義し, 利用者に対して適切なアクセスパスを提供することが可能となっている.

4 おわりに

本論文では, 本格的材料データシステムの実現に向けて, メタデータの表現とその実装例について述べた. メタデータの管理とこれに基づくデータの変換, ネットワーク上でのデータの位置の検索は異種の材料データベースを統合するために重要である. また, これによって材料データを利用するための利用目的に応じたアクセスパスを提供することが可能となる. シリーズ論文として, "材料データシステムの統合化(3)" において試作システムによるデータ検索例を示す予定である.

文 献

1) Viktor E. Hampel, William A. Bollinger, Carol A. Gaynor and James J. Oldani, An Online Directory of Database for Material Properties, Prepared for presentation to the Ninth International CODATA Conference(May 1984).

2) John Rumble, Joan Sauerwein, Sharon Pennell, Scientific and Technical Factual Database for Energy Research and Development Characteristics and Status for Physics, Chemistry, and Materials, DOE/TC/40017-1(Oct 1986).
3) H. Krokel, K. Reynard and G. Steven, Factual Material Data Bank, CEC Workshop - JRC Petten (Nov 1984).
4) Grattidge, Westbrook, McCarthy, Northrup and Rumble, Materials Information for Science & Technology (MIST): Project Overview, NBS Special Publication 726, U. S. Dept. of Commerce (Nov 1986).
5) J. G. Kaufman, The National Materials Property Data Network, Inc. - A Cooperative National Approach to Reliable Performance Data, ASTM STP101755-62(1989).
6) Commision of the European Communities D. G. XIII, TOWARDS A MATERIALS INFORMATION SERVICE FOR EUROPE, THE CEC MATERIALS DATA BANK DEMONSTRATOR PROGRAMME, F. L. A. Consultants
7) 芦野俊宏, 岩田修一材料データシステムの統合化(1)プロトタイプの設計とデータ型の拡張, 情報知識学会誌 1(1)75-91(Dec 1990).
8) V. M. Markowitz, S. Lewis, J. McCarthy, F. Olken, and M. Zorn, Data Management for Genomic Mapping Applications: A Case Study Proceedings of the 6th International Conference on Scientific and Statistical Database Management(To be published)
9) 三島良績, 岩田修一編, 新材料開発と材料設計学, ソフトサイエンス社(1985).

(1993年6月17日受付)
(1994年8月10日採録)

著者紹介

芦野俊宏
1962年生.1985年東京大学工学部原子力工学科卒業. 1990年同大学大学院博士課程修了. 工博. 同年新日本製鐵(株)入社. 現在同先端技術研究所主任研究員. 材料データベースに関する研究に従事. 材料設計のための計算機システム統合化に興味を持つ. 日本金属学会会員.

岩田修一(正会員)
1948年生.1975年東京大学大学院博士課程修了. 工博.1978年同大学工学部講師.1980年同助教授. 1985年より1年間, 西独 Fachinformationszentrum-Karlsruhe 客員研究員.1992年東京大学工学部教授. 現在同大学人工物工学研究センター, 工学部システム量子工学科教授を併任. 卒業より一貫して材料設計のための計算機システムの研究・開発に従事. 核融合炉材料, 核燃料を具体的な対象としつつ, 材料開発全般に対する普遍的な手法を追及中.