情 報 シ ス テ ム 37－4
（1992．1．21）

材料データベースの統合利用

芦野 俊宏
新日本製鉄(株) 技術開発本部 先端技術研究所

その所在が分散し、使用目的の多岐に渡る材料データベースを統合利用・運用するための分散型システムのプロトタイプについて述べる。データの所在や属性を表すメタデータを管理するサーバを導入することにより、異なったデータベース管理システムをネットワーク透過的にアクセスすることを可能とするとともに、データの変換やデータベースとのコネクションの設定を自動的に行なう機能を実現した。また、材料データの表現に必要とされる画像データや実験式などの手続きをファクトデータとともに統一的に扱うための枠組を開発した。

Integration of Material Database System

Toshihiro Ashino
Nippon Steel Corporation. Technical Development Bureau.
Advanced Materials & Technology Research Laboratories.
1618 Ida, Nakahara-Ku, Kawasaki 211, Japan

Design of computerized data system which integrates distributed and different material databases is discussed. This system enables transparent access to different database management systems. It supports automatic database connection and data conversion with meta-data which represents position of data on network and its attributes. Furthermore, data type extention needed to represent material data, such as image, procedure is discussed.

1 はじめに

設計を行なう際には、材料の特性や性能に関するデータが必要である。また、材料の試験データを用いて統計解析などの処理を行なうことによって材料の製造プロセスや成分の改良に役立てることが出来る。このため、データ検索や解析を目的として、計算機可読な材料データベースの開発が各処において行なわれているが、材料は種類が多く、かつ、各々の材料について製造方法、特性、用途、組織などの多様な情報を扱う必要があるために、データベースの目的や設計者の視点により、多種多様なデータモデルやデータベースマネジメントシステム (DBMS) が用いられている。[1, 2, 3]

また、本来同一の物性値であっても標準がないために、異なった名前や単位で表現されている場合がありうる。このため、エンドユーザが利用する材料データベースの所在やデータ構造を把握していることが必要となり、ネットワークが発達した今日においても材料の研究者や材料のユーザが複数のデータベースを用いる上での障害となっている。

さらに、材料データの表現には、一般的な商業用の DBMS ではサポートしていないデータ型を必要とする場合が多い。例えば、数値データを考えても、実験データの表現には範囲を持つ値、誤差、有効桁数などの情報を明示的に扱えることが望ましい。また、組織などを考える上では、写真などの画像データが重要である。

そこで、本論文では、上記の問題の解決するために、

- 異種のデータベース間での統一的なアクセス
- 材料データの表現のために必要な拡張されたデータ型の実現
- データの所在、属性といったメタデータの管理と利用

という 3 つの機能を実現するための分散型システムの設計とプロトタイプの作成について述べる。

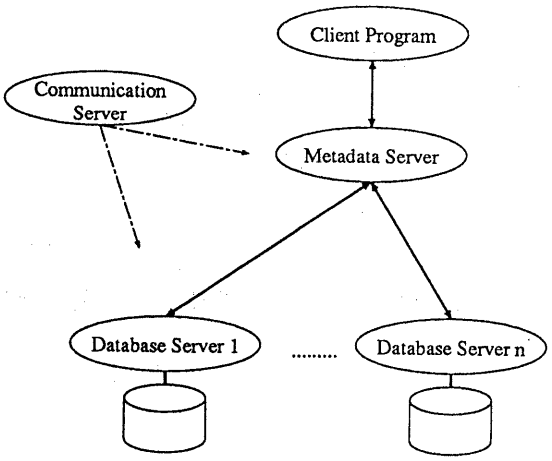

図 1: システム構成

2 システム構成

システム設計の基本としては、サーバ／クライアントモデルを用い、構成は図 1 に示したように 3 種類のサーバプロセスとクライアントプログラムとからなる。

各々のサーバプロセスの役割は、

Communication Server(CS) 通信ポートおよびアクセス権限の管理

Metadata Server(MS) メタデータの管理およびデータベースサーバへのコネクション設定、データ変換

Database Server(DS) 仮想化したデータベースサービスのネットワークへの提供

のように分けられている。以下、各々の実装について述べる。

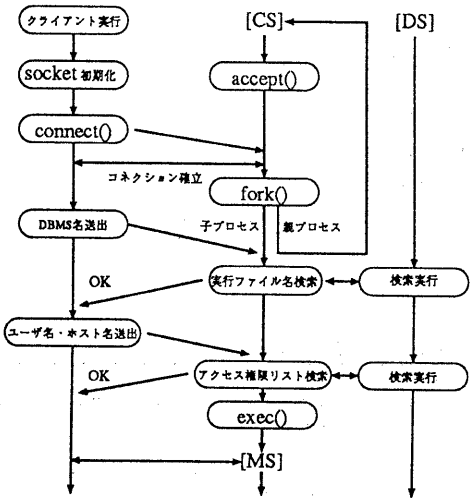

図 2: CS の動作

2.1 Communication Server

CS は関連する計算機上に常駐するプロセスであり、通信ポートを監視する。この意味では 4.3BSD UNIX における inet daemon と同様の動作をする。CS では、この他にユーザのアクセス権限の制限を行なう。クライアントが CS に接続要求を行なう場合の動作を図 2 に示す。

CS は表 1 に示すようなテーブルを持っており、これをユーザリストと呼んでいる。

クライアントは、起動時に環境変数 MDSERV にセットされたホストの CS へコネクションを設定する。このあと、クライアントは、接続したい DBMS の名前を送信する。一般に、クライアントが直接に交信するのは図 1 にあるように MS であり、これは “mdserv” という名前でユーザリストに登録されている。CS は DBMS 名に対応する実行ファイルの名前を検索するが、この検索は CS をコンパイル時に設定したホストの DS に対して行なわれる。このため、ユーザのアクセス権限は一か所のホストで集中管理することが可能になる。また、MS から DS への接続は自動的に行なわれるが、これも図 2 と同様に行なわれる。

2.2 Database Server

異なった DBMS への透過的なアクセスを可能とするためのベースとしては、SQL を用いた。しかし、ネットワークからの SQL によるアクセスのための標準プロトコルはシステム設計時に利用できなかったため、各々の DBMS に附属の埋め込み型 SQL を用いた。実際に利用した DBMS は、Ingres と ORACLE である。これらの埋め込み型 SQL においても、小さな点において相違があるため、これらの違いを吸収し、同一のプロトコルによるネットワークからのアクセスを可能とするため、表 2 に示すようなライブラリを作成し、仕様の違いをライブラリの内部で吸収させた。

DS は、このように対象とする DBMS に依存するライブラリとネットワークからの要求を受け、ライブラリを呼び出すメインルーチンとに分割されている。プロトタイプでは、データ検索を主目的においたため、表 2 に示すものしか実装していない。また、このなかで、db_record、Cursor_id とあるのは、各々データベースの一つのレコード、カーソルの id 情報を収めるためのデータ構造体である。db_record 内には、そのレコードにあるフィールドの数に対応した db_item という構造体がある。この定義を図 3 に示す。

図からわかるように、db_item は内部にデータ型を表す要素を持っており、データそのものは void ポインタによって指されている。本システムでは、全てのデータをこの形式で交換しているため、後に述べるようなデータ型の拡張を行なうことが容易になっている。

表 1: ユーザリストのデータ項目

フィールド名	内容
db	CS の常駐する計算機上にあるデータベース管理システムの名前
host	db で表される DBMS に接続を許されるホスト名
user	db で表される DBMS に接続を許されるユーザ名
executable	db で表される DBMS への接続を行なう DS の実行ファイルの名前

表 2: DBMS 依存ライブラリとそれに対応する SQL 文

ライブラリ	相当する SQL 文
int _db_connect(dbname); char *dbname;	EXEC SQL CONNECT dbname;
void *_db_select(field, table,where,group,order,record); char *field; char *table; char *where; char *group; char *order; db_record record:	EXEC SQL SELECT field INTO record->item[0].data..... FROM table WHERE where;
int _db_copen(did, field,table,where,group,order,cid); db_id *did; char *field; char *table; char *where; char *group; Cursor_Id *cid;	EXEC SQL DECLARE cid->name CURSOR FOR SELECT field FROM table WHERE where; EXEC SQL OPEN cid->name;
int _db_fetch(cid,record) Cursor_Id *cid; db_record *record;	EXEC SQL FETCH cid->name INTO record->item[0].data.....
int _db_cclose(cid) Cursor_Id *cid;	EXEC SQL CLOSE cid->name

```
typedef   struct    {
          u_char    type;       /* データ要素のデータ型 */
          char      *table;     /* 検索対象となっている表の名前 */
          char      *name;      /* データ要素のフィールド名 */
          int       len;        /* データ要素の長さ */
          char      indicator;  /* 標識変数 */
          void      *data;      /* データ本体 */
}         db_item;
```

図 3: データの内部表現

structure

label	*id*	*attr*	*type*	*unit*
Material Name	1	0	string	–

directory

host	*dbms*	*dbname*	*tname*
zodiac	oracle	generic	material

cname	*type*	*width*	*unit*	*id*
mname	string	30		1

dependency

key1	*key2*	*table1*	*table2*
id	id	material	property

図 4: メタデータ管理のための表

2.3 Metadata Server

MS は、図 1 に見られるように、クライアントの要求を解釈し、複数の DS に対して送信する。このとき、MS クライアントの要求に含まれるデータ項目名を図 4 に示したような三種類の表を用いて変換する。

本稿では、ここに示したような、ネットワーク上でのデータの所在およびその属性を記述するデータのことをメタデータと呼ぶ。各々の表の意味は、

structure 材料に関するデータ項目の一般的な名前と、システム内での id 番号、データ型、単位、上位階層の id との対応

directory structure で与えた id に対応するデータベース上でのフィールド名と属性および、そのデータベースの稼働しているホスト、DBMS

dependency あるデータベース上での二つのテーブルの対応関係

MS は、クライアントからデータの検索要求を受けた場合、最初に structure を参照し、id 番号を検索する。この id を用いて dicretory を検索することにより、該当するデータの存在する (ホスト名,DBMS 名, データベース名, テーブル名, フィールド名) という一組の情報が得られる。MS はこの情報に基づいて、必要な DS との接続がまだない場合には対応するホストの CS に要求を送り、接続を確立する。

検索対象のフィールドが同じデータベースの二つの表に分かれている場合には、dependency の情報を元に二つの表を結合した view を作成し、それを用いて以後の操作を行なう。

directory には、各々の項目の該当するデータベース上でのデータ型および単位の情報が記述されており、MS はこれ structure と DS から得られたデータを比較し、必要な場合には変換の後クライアントへ送り返す。材料データに必要な型の拡張は、このレベルで行なっている。

structure はまた、クライアントに対してアクセスできるデータ項目のメニューを提供するためにも用いられる。クライアントが MS に対して *label* のエントリを要求するためのプロトコルが用意されており、これをユーザに対してメニューとして提示することが出来る。このときに、*attr* フィールドを用いて図 5 に示す

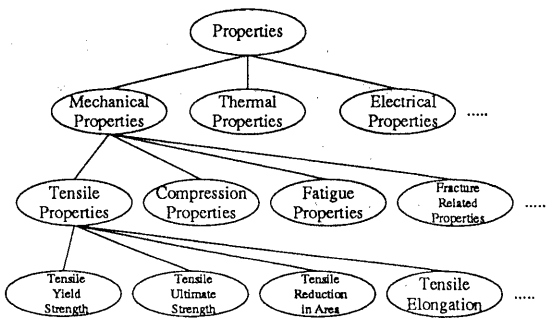

図 5: 材料特性の階層構造

ような木構造を作る。

図 5 は材料の特性を一般的なものから特殊なものへと辿った木構造の一例である。ここで出来る木構造は、単一の上位 id しか許さない単純なものである。しかし、データを一般的なものから特殊なものへ絞り込むための検索パスを提供するためには有効であるものと考えられる。structure, directory, dependency として用いる表はクライアント起動後に切り変えることが可能であり、ユーザの種類によって用いる検索パスを変更することが出来る。例えば、オーステナイト系、フェライト系など、材料の種類による検索パスを用いる、あるいは、システム管理のためには、データベースのあるホスト名、DBMS 名などで検索パスを作り、目的に応じて使い分けることが可能である。

2.4 プロトコル

MS⟺ DS および MS⟺ クライアント間での通信に用いる通信パケットの構造を図 6 に示す。

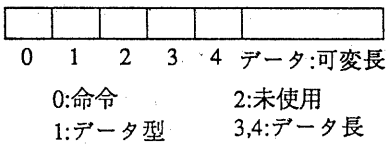

図 6: 通信パケットの構造

SQL の命令は、図に示すように {命令、(データ型)、(データ)} という形式に分解される。例えば、

```
SELECT ys,el
FROM   alloy_mks
INTO   ys,el
WHERE  alloy='MB1'
```

という命令は、

```
SELECT,STRING,"ys,el"
FROM,  STRING,"alloy_mks"
INTO,  FLOAT, "ys"
INTO,  FLOAT, "el"
WHERE,STRING,"alloy='MB1"'
```

のように分解される。数値データに関しては、一度 ASCII 形式に変換した後にデータを交換する。

3 データ型

材料データは一般に複雑な構造を持っており、画像などの商業用の DBMS で扱うことの出来ないデータが重要な位置を占める。[4, 5] このため、本システムでは、

- 誤差、範囲などを明示的に表現できる数値データ
- 属性を持ったビットイメージとしての画像データ
- 経験式、数値シミュレーションコードなどの手続き

の三種類のデータ型をサポートしている。実際には市販の DBMS パッケージを用いているため、これらのデータ型は MS のレベルで実現されている。すなわち、DS のレベルで用いられるデータ型は一般的な文字列、整数、浮動小数点数であり、DBMS 自体に変更はしない。このため、クライアントからの拡張データ型検索要求を複数のフィールドへの検索に変換している。以下、各々のデータ型について述べる。

3.1 数値データ

材料データを DBMS によって扱おうとする場合に生じる困難の一つは、数値データにお

いて範囲や誤差を明示的に扱うための方法が用意されていないことである。例えば、材料に関する文献には、

≤ 10

$400 \sim 500$

~ 0.01

のような表現が一般的に見られる。このために、一つの数値データに対して、

上限値	浮動少数点または整数の数値 (upper)
下限値	浮動少数点または整数の数値 (lower)
代表する値	浮動少数点または整数の数値 (val)
誤差	浮動少数点または整数の数値 (error)
有効桁数	整数の数値 (sig)
符号	文字 (sign1,sign2)
表記	文字列 (replesentation)
単位	文字列 (unit)

という情報を明示的に扱うことができるよう、MDB_INTEGER、MDB_FLOAT という二つの数値データ型を作成した。クライアントからこれらのデータ型の作成・検索要求があった場合、MS は上記の付加的なフィールドを付け加えた形式で DS へ要求を行なう。

また、クライアントからこれらの情報を明示して扱うために、SQL に若干の拡張を加え、

SELECT *table1.ys*
WHERE *table1.ys.error* < 0.1

のようにドットを一つだけ増やすことによって、誤差などの値を指定することとしている。また、何もつけずに指定した場合には、代表する値 (val) を返すようにすることにより、一般の SQL との互換性を保つようにしている。

材料データの検索の場合には、このようなデータに関して異なった扱いを要する場合がある。例えば、

SELECT *table1.ys,table1.material*
WHERE *table1.ys* > 50

という検索は ys が 50 以上になる材料名を要求している。このような検索を行なう場合、*table1.ys.val* が 50 以上の値になる材料名だけではなく、*table1.ys.upper* あるいは *table1.ys.val* +*table1.ys.error* が 50 以上の値になる材料名も得られることが望ましい。このようなルールは MS に組み込むことが可能である。

3.2 画像データ

材料を用いた実験では、組織写真をはじめとする画像データが重要な情報である。また、電子顕微鏡写真上のある領域の組織の名前や、組成分析値などをデータベースとして扱うための枠組が必要である。例えば、図 7 に示したようなデータ構造を表現することが出来れば画像データの検索、あるいはそれをキーとして組成分析や実験結果を扱うことが出来る。

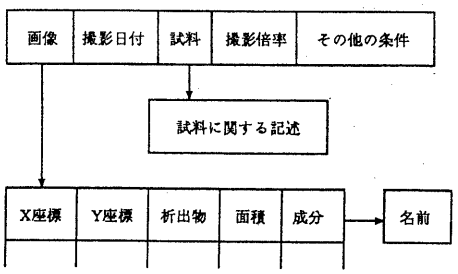

図 7: 電子顕微鏡写真のデータ構造

このようなデータを扱うために、数値データと同様に、ビットイメージとしての画像データを代表値としてそれに属性を持たせている。すなわち、ある日、ある倍率で撮影した画像のビットイメージを検索するには、

```
SELECT tem.image
INTO   :image
FROM   tem
WHERE  tem.image.date='yy/mm/dd'
   AND tem.image.magnification=20000
```

などと記述する。しかし、図 7 に示したような、析出物などの被写体に関するデータを表現しようとする場合、単に (x,y) 座標の組に対応する被写体のみではなく、図 7 のようにその名前や成分、面積のように二段階以上に重複した構造を持つために、この拡張のみでは表現しきれない。しかし、被写体の名前、または被写体に関して記述した表の名前は以下のようにして得ることができる。

```
SELECT tem.image.object
INTO   :object
FROM   tem
WHERE  tem.image.position=(20mm,20mm)
```

これ以上の、*object* の情報を入手するためには、非第一正規型やオブジェクト指向型のデータベースによって実現する必要があるが、ここでは、関係型データモデルを基本とするので、これ以上の拡張はおこなっていない。

DS のレベルでは、画像データは、ビットイメージを含むファイル名として DBMS 上にある。DS は画像データの検索要求を受けると該当するファイルの内容を MS へ送り、MS からクライアントへ送られる。このため、転送が二重になるが、これは画像データファイルがデータベースと同じ計算機上にあるものと想定したためである。

3.3 手続き

材料データを活用する上において、経験式、シミュレーションコード等の手続きをデータベースと同一のシステムで扱うことが出来た場合、

- データベースを手続きの入力データとして利用する
- 手続きの結果をデータベースにある実験データと比較することによって、手続き自体の改良・開発を行なう
- 手続きの結果をデータベースにある実験データと比較・補間することによって材料の開発、機器の設計に方向を与える
- データベースの内容に対して統計解析などの処理を行なう

といった応用が考えられる。このために、以下のような機能を実装している。

- データベースから検索した値を手続きの変数として与える機能
- 手続きの計算結果を、データベースの表と同様に扱う機能
- 手続き自体をデータベースと同様に検索する機能

手続きは、独立変数、従属変数、定数、検索用のキイワードという属性を持った一つのデータ型として扱われる。このようにみなすことにより、*procedures* という表が手続きを集めたものとすると、経験式 *function* に関しては、

```
SELECT procedures.function.y
INTO   :y
FROM   procedures.function
WHERE  procedures.function.x=0
          AND procedures.function.A=10
```

のように記述することによって、変数 x が 0、のときの y の値を得ることが可能になる。また、シミュレーションコード *proc* に関しては、

```
CREATE VIEW result(x,y)
WHERE AS
    SELECT    procedures.proc.x,
                  procedures.proc.y
    WHERE     procedures.proc.const1=0.1;
```

とすることにより、結果を表またはビューとして扱うことが出来る。このデータをクライアントから参照する場合は、一般的な表と同様にカーソルを設定し、これをフェッチして読み込むこととなる。手続きは、図8 に示したようなファイルに書かれており、ここからCのプログラムを合成、インクリメンタルローディングしたのち実行される。

4 応用例

図 9 にクライアントプログラムの例を示す。このデータは、材料の一般的な名称と代表的な特性値とを組み合わせたデータベースである。このようなデータベースは、材料の特性の全体的な様子を把握するために利用することが出来る。しかし、ここからさらに特定の材料の特性について詳しく知ろうとする場合には他のデータベースを参照しなければならないことが多い。本システムにおいては、一つのクライアントからさらに詳細なデータを検索することが可能である。

5 まとめ

材料データは現在の市販の DBMS パッケージで扱うには複雑な構造を持っており、数値計算、統計解析パッケージなどとの連携も不十分である。本論文では、この問題をネットワークを用いた分散システムによって解決する手法について検討を加えた。現在、データベースサーバに対する問い合わせプロトコルの標準化が進められており、より広範なシステムの統合化が可能であるものと考えられる。また、ユーザ定義データ型などの導入により、さらに汎用性の高いシステム構築が可能になるものと考えられる。

参考文献

[1] Carol A. Gaynor James J. Oldani Viktor E. Hampel, William A. Bollinger. An online directory of database for material properties. Prepared for presentation to the Ninth International CODATA Conference, may 1984.

[2] Sharon Pennell John Rumble, Joan Sauerwein. *Scientific and Technical Factual Databases for Energy Researchand Development - Characteristics and Status for Physics, Chemistry, and Materials.* DOE, oct 1986. DOE/TC/40017-1.

[3] G.Steven H.Krökel, K.Reynard. Factual material data bank. CEC Workshop – JRC Petten, nov 1984.

[4] 野上敦嗣. 材料データシステムに関する基礎的研究. PhD thesis, 東京大学, 1983.

[5] Philip M.Sargent. Materials data interchange for component manufacture. CUED/C-MATS/TR 147, Cambridge University Engineering Dept., sept 1988.

```
##
#title          Gilman-Johnston2_1962
#domain         single_crystal_Cu
#formula        dTau/dy=K/(2*A)-(2*b*Alpha/S(c))*(2*A/(K*L(0)))*((Tau/D)**m)
                *(K*y/(2*A)-Tau**2
#cformula       (K/(2*A))-(2*b*Alpha/S)*(2*A/(K*L))*(pow((double)(Tau/D),
                (double)m))*((K*y/(2*A))-Tau)*((K*y/(2*A))-Tau
##
##              name:meaning                    :unit
#i_variable     Tau:resolved_shear_stress       :g/(mm*mm)
#variable       y:crosshead_displacement        :1/10000in
##
##              name:meaning                    :range          :unit
#parameter      m:parameter                     :13_30          :no-dimension
#parameter      Alpha:parameter                 :1.0E08_1.0E10  :disl/(cm*cm)
##
#constant       D=540:parameter                 :5.4E02         :g/(mm*mm)
#constant       b=0.000000029:Burgers_vector    :2.9E-08        :cm
##
#keyword        single_crystal
#keyword        Cu
##
#assumption     Only pure edge and pure screw dislocations are present
                in a crystal and they move with different velocities
#assumption     n is nearly equal to n(s)
```

図 8: 手続きファイルの例

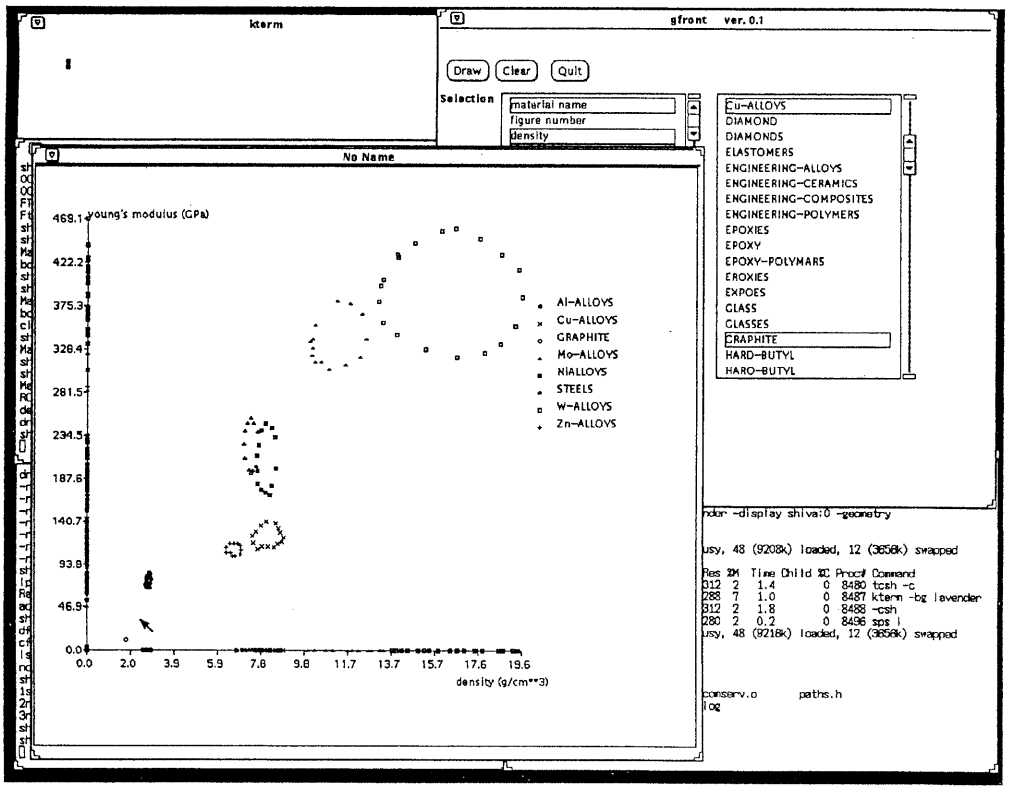

図 9: クライアントプログラム画面