

PostgreSQL をデータベースとしたサーバー & クライアント型 試薬管理システム (Servo) の開発

末永 正彦

九州大学理学研究院化学部門, 〒 812-8581 福岡市東区箱崎 6-10-1

e-mail: alohascc@mbox.nc.kyushu-u.ac.jp

(Received: July 26, 2002; Accepted for publication: November 25, 2002; Published on Web: January 31, 2003)

PostgreSQL をデータベースサーバーとし、データの入出力を行うクライアントを Java 言語で作成することで、サーバー & クライアント型の試薬管理システムを開発した。試薬瓶に付けたバーコードで管理を行い、データ入力の軽減を図るため天秤の読みの自動入力化とバーコード読み取り機を備えたシステムにした。サーバー & クライアント型の管理システムにする利点は、ネットワークを通してデータの授受ができる点にある。データベース本体と入出力部が分離できるので、複数の端末がある場合でもデータベースは一つでよく、試薬データの一元管理が容易になる。

キーワード：試薬管理システム, PostgreSQL, Java

1 はじめに

大学における試薬管理は、これまで企業におけるそれと比べてたいへん遅れていた感があるが、一連の毒物事件により試薬管理の見直しが社会的に要求されることになり、本大学でもそれに応える形で施設のできる試薬庫が導入された。それを機に、本研究室でも試薬のリストと各試薬の使用記録簿の作成をパソコンを利用して行うことが検討されたが、市販の試薬管理システムが高価であるため、試薬管理システムを Java 言語と PostgreSQL により自作した。

2 市販のシステムとの比較

市販の試薬管理システムである東北緑化環境保全株式会社の IASO 2000 [1] と本システムを比較してみる。バーコードと電子天秤とパソコンを連動させたシステムでネットワークを通じてデータのやり取りを行い、試薬管理のデータベースを一元管理しているシステムであるという点では同じであるが、IASO 2000 は Windows 上のクライアントであるのに対し、本システムは Java で記述されているため、電子天秤の接続など

一部の OS に依存する機能を除いて、プラットフォームに依存しない。ただし、試薬データの確認に関して、IASO ではクライアントとは別に Web ブラウザ - を使うこともできる。また、試薬の使用状況・履歴についても「いつ」「誰が」「どの薬品を」「どんな目的」で使用したかを記録する点でも GUI は別にして基本的に同じである。異なる点としては、以下のものが挙げられる。

IASO 2000 では、試薬の基礎情報（メーカー、使用期限、毒劇物か否か、構造式、物性など）を試薬ラベルのバーコードから読み込み、データベースにあるこれらのデータを表示させ、試薬管理用のデータベースにもそれがそのまま使えるが、本システムにはその機能がない。

IASO 2000 では、試薬の重量の異常増減などをクライアントの画面上の該当箇所に赤く点滅させるアラーム機能があるのに対し、本システムでは実装していない。これは、異常増減には登録のし忘れや間違った値の入力などユーザーの誤操作によるものが多く、これらを本当の異常と区別するのが難しいため、異常を判定する基準をまだ決めていないという理由による。

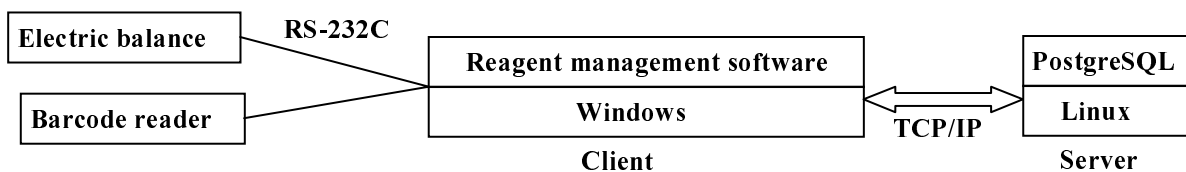


Figure 1. System diagram

また IASO 2000 では、電子式試薬保管庫（ナガノ科学機械製作所）、入室管理システム（富士通電装）、鍵管理システム（株式会社クマヒラ）と IASO 2000 を組み合わせたセキュリティシステムをオプションとして用意している。試薬管理を厳密に行う場合、各ユーザーの申告制による試薬の出し入れの記録では不十分で、試薬庫の開閉そのものを記録したり、試薬室への入室を記録するようなセキュリティシステムが必要となる。このような機能は、本システムにはない。

結論として本システムは、セキュリティシステム以外の機能では、ほぼ同じ機能を有するので簡易的な試薬の管理では十分利用できると思われる。

3 管理システムを自作する利点

試薬管理システムのようなデータベースクライアントは、データベースの構造（テーブルの数や、テーブルを構成するフィールドの数やその属性）に強く依存しているため、データベースの構造を変更した場合（例えば、新たなフィールドを追加した場合）、クライアント側も書き換えることが必要になる。市販のシステムで用意されているデータベースの仕様に、ユーザーが望む必須項目がない場合、システムを書き換えを依頼することになる。このようなカスタマイズの要望は各社有償で受け付けているようであるが、自作している場合、このような変更を簡単に素早く行うことができる。また、データベースの構造の変更ではなく、データベースそのものを PostgreSQL から別のデータベースに取り替える場合も簡単に行うことができる。すでに別の目的で運用している PostgreSQL 以外のデータベースを用いて試薬の管理もいっしょに行おうとする場合でも、移行は容易である。このほかクライアントの使い勝手の細かな変更なども、自作している場合には自由にできるので、利用者がより使い易いと思われるシステムに仕上げていくことができる。

このようにシステムを自作した場合、システム仕様

の変更、拡張を自由に行うことができるのが最大の利点である。

4 システム構成

本試薬管理システムは、Figure 1 に示すように、データベースを管理するサーバー部分とデータベースへの入出力インターフェイスとなるクライアントからできている。データベースは RedHat Linux 6.1 上で稼働する PostgreSQL 6.5.1 を使用した [2]。クライアントは Java 言語で新たに開発し、Windows 上で稼働している [3]。システムをサーバー&クライアント型としたことで、データベースを一元化することができ、同時に多くのユーザーがデータベースのあるマシンとは別の離れたところにあるマシンからネットワークを経由して利用することが可能になった。実際の試薬の管理はそれぞれの試薬瓶に貼ったバーコードをもとに、試薬瓶の重量も含めた全量の変化を記録して行う。バーコード入力軽減のため、クライアントマシンにはバーコードリーダーがつながっている。また、電子天秤もシリアルポートを介して RS-232C でつながっており、試薬瓶の重量の入力が自動化されている。

5 クライアントの概要

クライアントは目的別（登録、使用、返却、試薬検索、履歴検索、破棄、シリアルポートの開閉、データベース保守）に独立した 8 個のフレームとサーバーからの応答を表示する 2 個のフレームおよび全フレームの表示を切り替えるコントロールフレーム（Figure 2）から成っている。各フレームの GUI は Java Swing [4] を用いて作成したが、Swing の Look&Feel とは若干異なり、画像を多く使用した独自のものに仕上げた。またフレーム毎に背景の色を変え、どの操作をしているのか直感的に区別できるようにした。

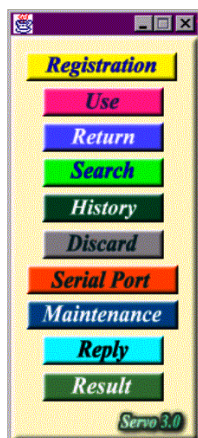


Figure 2. Control frame

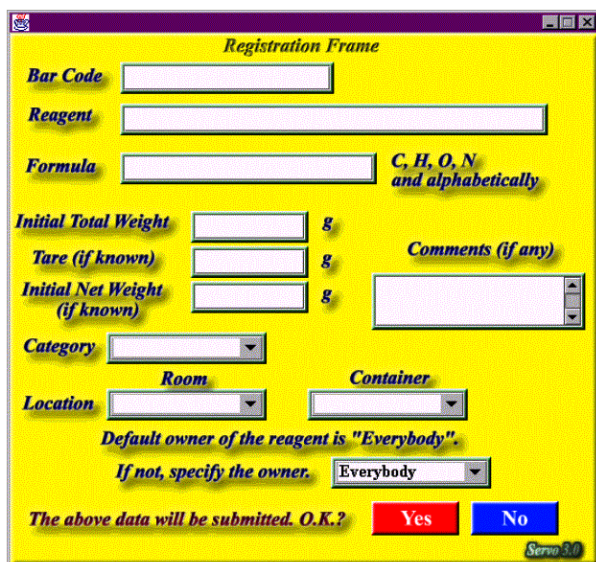


Figure 3. Registration frame

以下、コントロールフレームにあるボタンの順に、各フレームを簡単に説明する。

Figure 3 からわかるように、バーコード、試薬名、化学式、瓶を含めた全体の重さを入力し、試薬の分類 (Category) 部屋、試薬庫をリストから選択して、試薬を登録する。試薬の所有者は、デフォルトでは「全員」になっている。また、試薬瓶の風袋、試薬の正味の重量は、不明な場合は入力しなくてもよい。コメントも必須ではない。登録に使うバーコードはデータベースの中では整数として取り扱っているので、数字で表される規格のものであればよい。また、試薬の分類 (Category) は、「毒物及び劇物取締法」による区

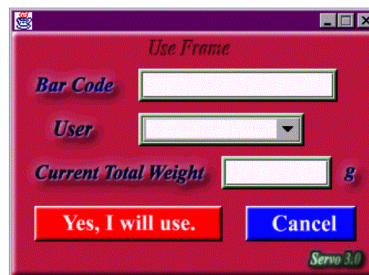


Figure 4. Use frame



Figure 5. Return frame



Figure 6. Reply frame

分に従い、劇物 (Hazardous) 毒物 (Toxic) その他 (Others) の3つとしている。

試薬の使用、返却は、それぞれ別のフレーム (Figures 4, 5) を使い、必要最小限のデータだけが表示されるようにしてある。試薬を返却する際のユーザー確認は、現在のところ行わない仕様にしており、履歴には、「Last User が返却した」と記録される。

使用、返却に伴うデータベース更新の際に PostgreSQL から返されるメッセージや不適当なデータの入力に対するエラーは、Figure 6 に示されるフレームにすべて表示される。

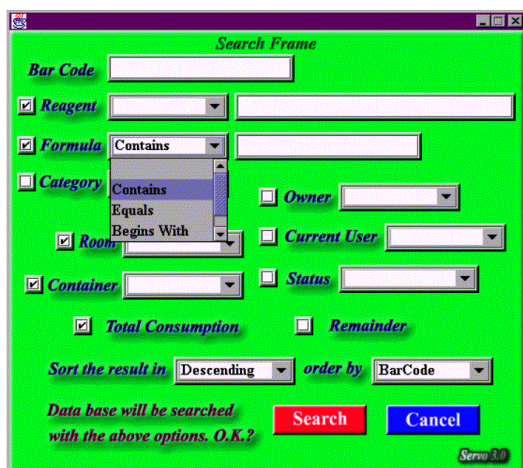


Figure 7. Reagent search frame

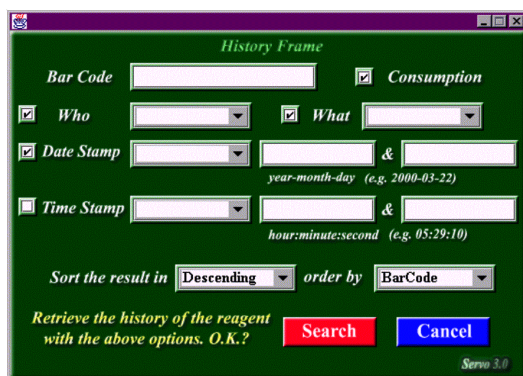


Figure 8. History search frame

試薬の検索は Figure 7 に示されるようなフレームを使う。検索の条件は各項目に続くテキストフィールドに条件を書くことにより設定される。空白は「条件設定なし」の意味になる。従って、何も入力せずに検索させると全ての試薬が表示される。また、条件となる文字列は大文字と小文字が区別されるため、例えば Sodium で登録してある試薬を sodium で検索してもヒットしない。そのため、登録作業を大人数で手分けして行う場合、試薬名は「頭文字だけ大文字にする」といったような規則を作り試薬名の表記法を統一したほうがよい。また検索画面のチェックマークの有無は、その項目を検索結果のなかで表示するかどうかという意味であり、その項目で条件を付けるかどうかという意味ではない。検索結果の並べ替えはデフォルトでは、バーコードの大きい順にしてあるが、他の項目でも並べ替えができる。

試薬の履歴は、Figure 8 のようなフレームで行う。検索における条件の入れ方やチェックマークの意味は、

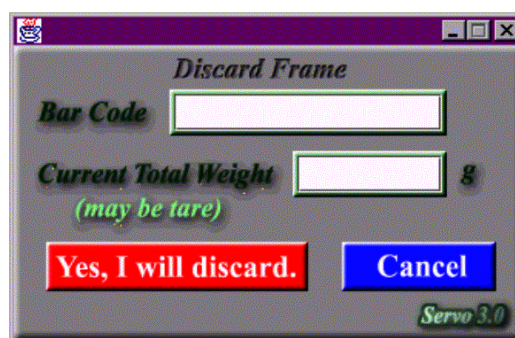


Figure 9. Discard frame

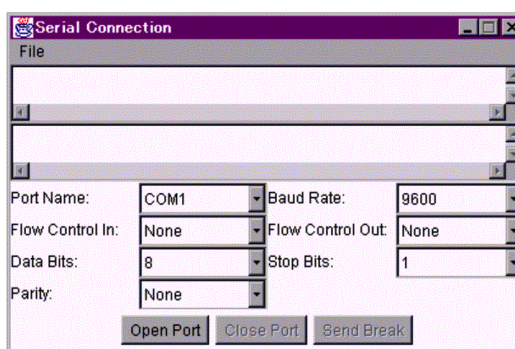


Figure 10. Serial connection frame

試薬検索で述べたのと同じである。特定の試薬の全ての履歴(登録、使用、返却、廃棄等)をみるには、バーコードのみを入力し、検索させる。何も入力せずに検索させると全ての試薬の全ての履歴が表示される。日時、ユーザー名や登録、廃棄などのイベントによる絞り込み検索もできる。

試薬の廃棄手続は Figure 9 に示すフレームで行う。機能的には Figures 4, 5 に示した試薬の貸出しや返却と同じだが、どの手続をしているのか明確にし、間違いが少なくなるようにするため、これらは別々のフレームにしてある。

シリアルポート設定、開閉は Figure 10 に示すようなフレームで行う。これは、Java Communications API に付属の Demo に電子天秤との通信を行う部分を付け加えて作った。電子天秤の読みを表示させるためには、ポート名や Baud Rate 等を設定した上で、“Open Port” ボタンによりシリアルポートをオープンしておかなければならない。

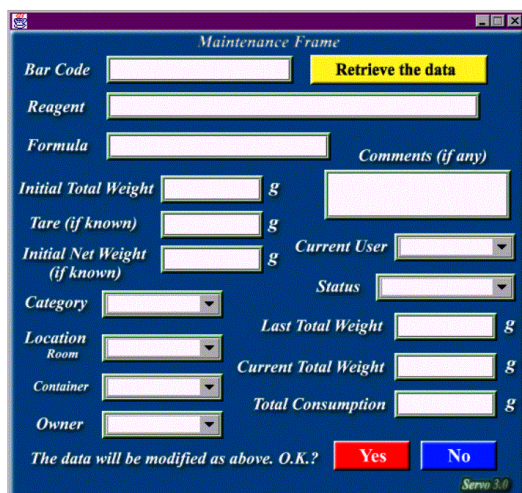


Figure 11. Maintenance frame

データベースへの誤入力をクライアント側で修正するためのフレームが Figure 11 に示されている。バーコードを入力し“Retrieve the data”ボタンをクリックすると相当する試薬のデータ（テーブル reagents のデータ）がすべて表示されるので、これを修正する。試薬の履歴（テーブル history のデータ）は、クライアント側では修正できない。データデータベースの修正はこのフレームでの作業のほか、7.3 で述べるようにテキスト化されたデータベースに対して行う方法もある。試薬の履歴の修正は、テキスト化されたデータベースを修正しなければならない。

試薬検索、履歴検索のフレームでチェックマークを入れた項目の検索結果は、Figure 12 のようなテーブルの形で表示される。この表のそれぞれのカラムの幅は、項目を区切る縦線をドラッグすることにより変更することができる。また結果は、テキストファイルとして保存できる。

6 クライアントの設定

6.1 ユーザー名などの登録

保管場所、試薬庫の番号、ユーザー名は、Java ソースファイル Constants.java の中で定数 LOCATION1, LOCATION2, USERS として定義されているので、その登録はこれらの定数を設定することにより行う。

データベースの URL（あるいは IP アドレス）およびデータベースにログインする際に必要なログイン名（上で述べたユーザー名とは別）とそのパスワードは、

| No | barcode | reagent | formula | location1 | location2 | totalconsum |
|----|----------|--------------------|------------|-----------|-----------|-------------|
| 1 | 49626063 | Instant Coffee | XXXX | Lab2 | None | 0.0 |
| 2 | 10009918 | Glycine | H2NCH2... | Room2306 | No.12 | 0.27 |
| 3 | 10009901 | Glycine | H2NCH2... | Room2306 | No.12 | 0.0 |
| 4 | 10009895 | Calcium Hydride | CaH2 | Room2306 | No.10 | 0.0 |
| 5 | 10009888 | Sodium Hydride... | NaH | Room2306 | No.10 | 0.0 |
| 6 | 10009871 | Potassium Hydr... | KH | Room2306 | No.10 | 0.0 |
| 7 | 10009864 | Sodium Hydride | NaH | Room2306 | No.10 | 0.0 |
| 8 | 10009857 | Benzotrichloride | C6H5CCl3 | Room2306 | No.10 | 0.0 |
| 9 | 10009840 | Benzyl bromide | C6H5CH... | Room2306 | No.10 | 0.0 |
| 10 | 10009833 | Benzyl Chloride | C6H5CH... | Room2306 | No.10 | 0.0 |
| 11 | 10009826 | Benzoyl Peroxide | (C6H5CO... | Room2306 | No.10 | 0.0 |
| 12 | 10009819 | 1,1,1-Trimethox... | CH3C(OC... | Room2306 | No.10 | 0.0 |
| 13 | 10009802 | Phenyl Isocyan... | C6H5NCO | Room2306 | No.10 | 0.0 |

Figure 12. Result frame

7.4 の「データベースのセキュリティ」で説明してあるログイン名とパスワードのことで、ソースファイル DBConfigure.java の中で定数 URL, USER, PASSWD として定義されている。その登録はこれらの定数を設定することにより行う。

以上の登録作業は、Java のソースファイルを変更するので、使用するためには再コンパイルが必要となる。

6.2 管理ユーザーのパスワードの登録

本システムでは、管理ユーザーというものを特には設けていない。ただ、Figure 11 で示したデータベースの修正フレームを表示する際、パスワードによる制限を設けているだけである。したがってこのパスワードが実質的な管理ユーザーのパスワードということになる。このパスワードは Java ソースファイル Constants.java の中で定数 SUPERPASSWD として定義されている。ただし、パスワードの文字列がそのまま入る訳ではなく、MD5 によりダイジェスト化された 32 桁の 16 進数が入っている。ダイジェスト化は、例えば Linux 上の md5sum コマンドにより行う。

この登録作業もまた、Java のソースファイルを変更するので、使用するためには再コンパイルが必要となる。

6.3 JDBC ドライバ

Java から PostgreSQL を使うためには、JDBC ドライバが必要である。注意しなければならないことは、Java コンパイラと PostgreSQL の双方のバージョンに適合したドライバーを使う必要があるということである。下記の JDBC のサイト

<http://jdbc.postgresql.org/download.html>

に適切なドライバーの圧縮アーカイブが置いてあるので、ダウンロードして使用する。ちなみに、著者が使用した JDK 1.1.8 と PostgreSQL 6.5.1 の組合せでは、ダウンロードのページにある PostgreSQL 6.5.2 JDK 1.1.x を選択して、jdbc6.5-1.1.jar をダウンロードした。この圧縮アーカイブを解凍して得られる postgresql というフォルダーは、フォルダーごと Servo の Class ファイルのあるフォルダー内に置いておかなければならない。

6.4 周辺機器

天秤の読みを自動的にパソコンに取り込むため、天秤とクライアントマシンが RS-232C で接続されている。通信手段としては MT-SICS (METTLER TOLEDO Standard Interface Command Set) を用いている。MT-SICS は、METTLER-TOLEDO 社が開発した電子天秤を制御するためのコマンド群である。本システムで使用している MT-SICS のコマンド S は「安定している現在の重量の値を送信せよ」という命令である。“S”という文字を二秒毎にシリアルポートを介して天秤側に送信することにより、送り返される文字列の中から重量を表す数値を取り出して表示している。MT-SICS 以外の通信手段を使用する場合には、天秤から返されるデータのフォーマットに応じて若干の変更が必要となる。Java からシリアルポートを使うためには、Java Communications API が必要となる。これは、

<http://java.sun.com/products/javacomm/index.html>

からダウンロードできるが、対応している OS は、現在のところ Windows と Solaris だけである。したがって、クライアントを Macintosh 上で稼働させる場合、シリアルポートに関する部分を削除しておかなければならない。

このほか、バーコードの入力を軽減するためバーコードリーダーが接続されている。用いたのは、オプトエレクトロニクス社のキーボードインターフェイスバーコードスキャナ (OPT-WEDGE シリーズ) であり、これはパソコン本体とキーボードの間に接続するだけ

で使用できる。

7 サーバーの設定と保守管理

7.1 データベースの構成

データベースは二つのテーブル (Reagents と History) で構成されている。テーブルの定義は、PostgreSQL のコマンドの形で示すと List 1, 2 のようになる。テーブル Reagents では、すべての試薬の現時点での状態だけを記録する。したがって一つの試薬にただ一つのデータがある。そのため Barcode の項目には重複がないよう primary key が指定してある。対照的に、テーブル History では各試薬の履歴を全部記録するため、一つの試薬に履歴の数だけデータがある。従ってテーブル History の Barcode のデータは重複があってもよい。「not null」が指定してあるものは、空白が不可な項目である。また CurrentTotalWeight のように常に 0.00g 以上であるべきような項目は、check の後に続く括弧内の条件に適合するかどうかが、更新時にチェックされる。TotalConsumption は 0.00g の場合もあるので、-0.01g を下限としてチェックをしている。

```
CREATE TABLE Reagents (  
Barcode          int          primary key,  
Reagent          text        not null,  
Formula          text        not null,  
InitTotalWeight  real        not null,  
Tare             real,  
InitNetWeight    real,  
LastTotalWeight  real  
                not null check (LastTotalWeight > 0.00),  
CurrentTotalWeight real  
                not null check (CurrentTotalWeight > 0.00),  
TotalConsumption real  
                not null check (TotalConsumption > -0.01),  
Category         text        not null,  
Location1        text        not null,  
Location2        text        not null,  
Owner            text        not null,  
CurrentUser      text        not null,  
Comment          text,  
Status           text        not null  
);
```

List 1. テーブル reagents の定義

```
CREATE TABLE History (  
Barcode          int          not null,  
LastTotalWeight  real        not null,  
CurrentTotalWeight real  
                not null  
                check (CurrentTotalWeight > 0.00),  
Who              text        not null,  
DateStamp        text        not null,  
TimeStamp        text        not null,  
What             text        not null  
);
```

List 2. テーブル history の定義

7.2 データベースの作り方

データベースは、Linux 上で、PostgreSQL のインタプリタである `psql` を使い対話的に作成する。ただし、PostgreSQL が既にインストールされており、クライアントからの要求を受け付ける `postmaster` というデーモンが起動しているものとする。

まず、List 1, 2 にある二つのテーブルの定義をそれぞれテキストファイル `reagents.sql` と `history.sql` に保存する。Linux のシェルから `createdb servodb` と入力し、`servodb` という名のデータベースをつくる。次にシェルから `psql servodb` と入力し、`psql` を起動する。`psql` のプロンプトがあるので、`¥i reagents.sql;` と入力する。同様に `¥i history.sql;` と入力する。これで、二つのテーブルが新たに作成される。生成したテーブルの確認は、`psql` のプロンプトから `¥d reagents;` あるいは `¥d history;` を入力して行う。最後に `¥q` と入力し、`psql` を終了する。以上の操作で二つのテーブル `Reagents` と `History` を持つデータベース `servodb` ができる。

7.3 サーバー側でのデータベースの保守

データベースをテキストファイルとして残すには、`pg_dump` コマンドを使う。Linux のシェルから `pg_dump servodb > servodb.pgdump` を実行すると `servodb` というデータベースの内容が、`servodb.pgdump` というファイル名で保存される。このファイルはテキストファイルなので、データの削除や追加など編集ができる。`pg_dump` コマンドにより作成したテキストファイル化したデータベースを読み込み、データベースを復元するには、つぎのコマンドを Linux のシェルから実行する。

```
cat servodb.pgdump | psql servodb
```

テキスト化したデータベースを編集する際には、空白に関し `Space` と `Tab` の使い分けに注意する必要がある。例えば、次の様なデータベースがあった場合、

| ID | NAME | AGE |
|----|------------------|-----|
| 1 | Amidala | ¥N |
| 2 | Anakin Skywalker | 8 |

各項目 (`ID`, `NAME`, `AGE`) を分けている空白は `Tab` であり `Space` ではない。「Anakin Skywalker」のように空白を含む文字列では、空白としては `Space` を使い、`Tab` を使ってはならない。また、データがない項目には `¥N` が入っているが、この部分を空白にしてはなら

ない。空白の取り扱いを間違えるとテキストファイル化したデータベースを再び読み込んでデータベースを復元する際にエラーがでてうまくできない。

7.4 データベースのセキュリティ

データベースへのアクセス制限の条件は、`pg_hba.conf` に記述する。このファイルは PostgreSQL のインストール時に既に作成されている。この中の `host` の部分にアクセスの制限を記述する。例えば、
local all trust
host all 0.0.0.0 0.0.0.0 password passwd

と書くと、データベースのあるマシンからは何の制限もなくアクセスでき、ネットワークを介したアクセスはパスワードによる制限がかかる。ここで、`passwd` は PostgreSQL へ接続する時に必要なパスワードであり、シェルから `pg_pass` コマンドを実行することでログイン名とともに設定することができる。この `passwd` はクライアントがサーバーにログインする場合に使われるため、6.1 で説明した定数 `PASSWD` に設定する。

このほか IP アドレスによるアクセス制限も行うことができ、`0.0.0.0 0.0.0.0` の部分を例えば、
192.168.0.1 255.255.255.255 と書くと、
192.168.0.1 からのアクセスのみが許可される。

8 今後の改良点

現在のシステムは、ユーザー名、試薬庫などデータベースを実際に使用するにあたって必要な項目を Java ソースファイルの中で定数として定義しているため再コンパイルを必要としているが、設定に必要な項目はすべてテキストファイルとし、クライアントの起動時にそれを読み込むことで設定できるように変更するのがよいと思われる。また、本システムは、筆者の所属する研究室で使用するため作成したものであるため、パスワードによるアクセスの制限を非常に緩やかなものにしてあるが、今後は厳しい制限でも緩やかな制限でも、どちらにも簡単に対応してカスタマイズできるようなシステムにすべきである。

その他の問題点としては、PostgreSQL を動かしている Linux マシンがダウンした際に代替となる Linux マシンが用意できない場合、試薬管理システムがダウンするということが挙げられる。Linux マシンよりも研究室に多くある Windows マシンをデータベースサー

バーにすることができれば、代替マシンが見つめ易くなると思われる。また、試薬データベースを定期的に保存するなどの作業も Linux に慣れた管理者がいなくなった場合、行われなくなる可能性が高くなる。サーバーマシンが Windows であれば、このような問題も少しは軽減されると思われる。したがって、Windows 上でも動くデータベースをもとにした試薬管理システムを作ったほうが、マシンのクラッシュに強く、管理や保守のし易いシステムになると考えられる。候補となるデータベースとして MySQL が挙げられる。これは、高速で堅牢なマルチユーザー・マルチスレッドの SQL データベースであり、無償で利用できる。次のバージョンは、この MySQL をもとに作成する予定である。

9 ソフトウェアの配布

Servo (Ver.3.0) は Java ソースコードを含め全てフリーで著者のホームページよりダウンロードできる。

<http://hb6.seikyoku.ne.jp/home/zzzfelis>

参考文献

- [1] IASO 2000 のカタログ、東北緑化環境保全株式会社.
- [2] 石井達夫, *PostgreSQL 完全攻略ガイド*, 技術評論社 (1999).
- [3] George Reese 著、古賀直樹 監訳、川村史記、鶴町重夫 共訳, *JDBC による JAVA データベースプログラミング*, オライリージャパン (1998).
- [4] Satyaraj Pantham 著、岩谷 宏 訳, *速習 Java Swing プログラミング*, SOFT BANK Publishing (1999).

Development of a Server/Client Type Reagent Management System (Servo) Using PostgreSQL as a Database

Masahiko SUENAGA

Department of Chemistry, Faculty of Sciences, Kyushu University
812-8581 Hakozaki 6-10-1, Higashi-ku, Fukuoka Japan
e-mail: alohascc@mbox.nc.kyushu-u.ac.jp

A server/client type reagent management system was developed. The system is briefly illustrated in Figure 1. PostgreSQL 6.5.1 running on the RedHat Linux 6.1 serves as a database. As a client to the database server, reagent management software was newly programmed with Java language. In order to obtain the weight readings automatically, an electric balance is connected to the client machine via RS-232C. Additionally, the barcode reader facilitates barcode input.

In the system, each of the reagents is managed by a barcode labeled on the bottle. Since the system is server/client type, multiusers can access the database simultaneously from remote client machines and the maintenance of the database can be centralized.

The client software including the source code can be downloaded from the Web site of the author.

<http://hb6.seikyoku.ne.jp/home/zzzfelis>

Keywords: Reagent management system, PostgreSQL, Java