平成１２年度　技術研究会論文

Ｌｉｎｕｘ　ＤＢサーバの構築と評価

～Ｌｉｎｕｘサーバってどこまでやれるの？

大分県情報サービス産業協会
技術委員会　Ｌｉｎｕｘ部会

～　論文目次　～

１．はじめに

以前より、「Ｌｉｎｕｘ」を対象としたサービスや製品が相次いで発表されている事は、情報処理産業に携わる者のみならず、利用者と言う立場の者としても興味があるところではないだろうか。

「Ｌｉｎｕｘ」と言う言葉で連想されることとして、

(1).オペレーティングシステム(Ｌｉｎｕｘ)が低価格または無償で入手可能

(2).低スペック(遊休資産)のハードウェアでも利用可能

(3).稼動可能なソフトウェア(アプリケーション)も低価格または無償で入手可能

等が挙げられる。これらは、平成１１年度開催の同技術委員会中でのＬｉｎｕｘ部会がテーマとし、勉強会を通じて確認した内容である。

そこで、本部会ではＲＤＢＭＳ(リレーショナルデーターベースマネージメントシステム)に視点を置きＬｉｎｕｘをベースとしたＲＤＢサーバの構築と動作検証そして、その評価を行う事とした。

無論、Ｌｉｎｕｘの導入と調整等も必要になる事から、平成１１年度の技術委員会で発表された論文：『Ｌｉｎｕｘサーバー構築のガイドライン』を参考にしながら進めていく事とする。

２．メンバ紹介

部会長	髙橋　昭光	三井造船システム技研株式会社 九州事業所 カストマアプリケーション事業部　第三開発部
副部会長	西河原　洋一	株式会社 オーイーシー ソリューション営業部 ビジネスシステム営業課
メンバ	亀井　浩	株式会社 富士通大分ソフトウェアラボラトリ ソリューションシステム事業部 第二ソリューション開発部
	安井　正樹	株式会社 富士通大分ソフトウェアラボラトリ ソリューションシステム事業部 第一ソリューション開発部
	加藤　匡	鶴崎海陸運輸株式会社 システム事業部 システム開発課
	宗安　隆行	三井造船システム技研株式会社 九州事業所 カストマアプリケーション事業部　第三開発部
技術委員		
	糸野　憲和	株式会社 長嶋不動産鑑定事務所

３．活動経緯

3.(1). テーマの選定

本研究部会では、DB サーバとしての Linux を実機上で評価することをテーマとした。以下にそのテーマ選定の理由・過程を示す。

a. なぜデータベースサーバなのか？

今回、DB サーバとしての Linux 運用をテーマとして選定したのは、以下の理由による。

- やはりサーバ機 OS として Linux の実力を評価したい。Linux のデスクトップ機 OS としての機能は（ウィンドウマネージャやオフィスアプリケーションなど）充実しつつはあるが、率直に判断して、未だ Windows と比較して、評価するには尚早である。
- 文書サーバや WEB サーバとしての Linux 運用は、テーマとしての新規性がない。
- 部会メンバの多数がデータベースの適用業務に携わっている（あるいは経験がある）ため、Linux をデータベースサーバとした場合にどこまでのことができるのか、興味がある。

b. なぜ実機を用いて検証するのか？

文献等の情報にあたることで、Linux の DB サーバとしての実力や可能性を評価することも検討した。確かにその場合、より包括的な観点からの評価・まとめが可能であるだろうと考えられる。しかし、本部会では、実際に実機上で DB サーバとして運用してみることにした。なぜならば、単なる情報のまとめにとどまらない、以下のような現実的な観点からの運用性の判断をおこなうことを重視したためである。

- 実際に導入, 運用することで、実業務を想定した導入の留意点などを検証したい。
- われわれの想定した業務ケースで、データベースの性能を評価してみたい。

c. 何を検証/評価するのか？

Linux のメリットとして、要求される資源が比較的少ないため、低スペックの PC をサーバとして活用できることがあげられる。そこで、現実的な想定として、数万件～数十万件の小規模データベースを低スペック PC で運用し、その運用性, 安定性, 性能を評価することとした。

d. テーマをどのように絞り込んだのか？

Linux はサーバ用 OS として、WindowsNT (あるいは Windows 2000) Server の対抗 OS の有力候補のひとつである。Linux の実力を総合的に判断するには、WindowsNT との比較という観点が必要であるのはいうまでもない。しかし、本研究部会でそこまでテーマを広げるのは、期間的に不可能であると判断し、データベースサーバとしての評価にとどめることとした。

3.(2). 検証方法および比較対象の決定

先に記述したように、今回の検証の基本的な問題意識は、以下のとおりである。

> Linux と Linux 上で動作する RDBMS の組み合わせは、小規模データベースの実業務運用の解となりうるのか。

上記問題意識を前提とし、業務を想定したかたちで Linux の動作検証をおこなうため、以下の手順で作業をすすめることとした。

- Linux 上で動作するデータベースを作成・運用し、運用性や安定性を確認する。
- データベースを検索・更新するアプリケーションを作成する。
- 上記アプリケーションでの動作性能を測定する。
  ※市販データベースとの比較をおこなう。

具体的な検証方法および比較対象などの決定内容およびその根拠について、以下に示す。基本的には、入手が容易であること、および一般に普及していることを前提とした。

検証方法および比較対象

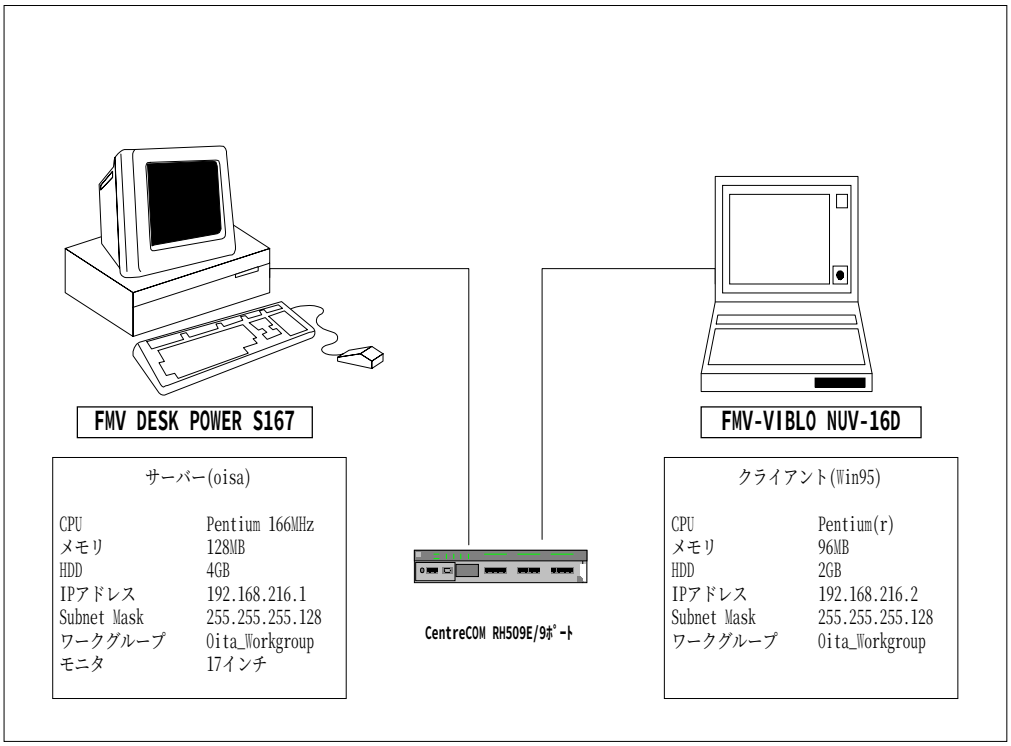
項目	決定内容	根拠
ハードウェア	CPU　：Pentium166MHz メモリ：48MB ディスク：4GB	部会メンバから提供できる遊休資産を利用する。遊休資産の再活用は、Linux の活用方法として一般的であるとされているが、現実に小規模データベース運用の場合に解となりうるのか、検証の意味があると判断した。
OS (Linux)	RedHat Linux 6.2J (Release 2.2.14-5.0)	Linux には様々なディストリビューションがあるが、最も普及しているディストリビューションであり、実際に部会メンバが入手済みであった RedHat Linux の最新版（7 月検討時点）を利用する。
DBMS	PostgreSQL（ver 6.5.3.）	PostgreSQL は Linux で動作するフリーソフトウェア DBMS の中で性能・安定性の面で評価が高く、普及率が高い。また、Linux のディストリビューションに標準添付されている。

項目	決定内容	根拠
データ	郵便番号データ	自由に利用できるデータであり、かつデータ件数も数万件で今回の検証の想定規模に合致すると判断した。
アプリケーション	Java アプリケーション（新規作成）	Java アプリケーションを用いた、ネットワーク経由のデータベース利用は、現時点でもっとも一般的な利用形態であると想定される。
クライアント PC	Windows9x 搭載マシン	サーバは Linux であっても、クライアントはWindowsであるのが一般的な利用形態である。
比較対象 DBMS	Oracle8i for Linux 試用版 (※)	Linux+PostgreSQL の可用性などを検証するために、最も普及率の高い市販RDBMS である Oracle の試用版を入手し、同様のデータベースの構築、検索・更新を実施する。

※ 今回用いた Oracle8i は試用版であるため、ベンチマークテスト結果を Oracle 社の許可なしに第三者に公開することは、ライセンスによって禁止されている。
そのため、今回の検証では、PostgreSQL と Oracle との性能比較は主眼としない。導入の容易さ，運用性，あるいは安定性などの指標として、Oracle を用いるものである。

3.(3). 実機構成

ハードウェア構成（最終構成）



3.(4). 導入作業

a. オペレーティングシステム

インストール環境は以下のとおりである。

マシン：FMV S167 (CPU:Pentium 166MHz, メモリ:48MB, HDD:4GB)
OS(Linux)：RedHat Linux 6.2J

Linux のインストールは GUI ベースのインストーラの指示にしたがっておこなう。ここではその操作の詳細については記述しない。インストール時に指定した項目についてのみ、以下に示す。

設定項目	内容
ディスクスライス構成	/ (2GB), /usr (1.8GB), swap (36MB)
ネットワーク構成	ホスト名 oisa IP アドレス 192.168.216.1 サブネットマスク: 255.255.255.128
一般ユーザの登録	takahasi, muneyasu, nisigawa, kamei, yasui, itono, katou
パッケージグループの選択 （カスタマイズ）	- Printer Support - X Window - GNOME - Network Workstation - SMB (Sumba) Server - Postgres (SQL) Server - Network Management Workstation - Utility

インストール後、OS に対していくつかの変更作業が必要であった。以下に列挙する。

作業 1：グラフィカルログインモードの解除

[背景] メモリ資源が不足しているため、インストール時の標準であるグラフィカルログインモードを解除し、X Window System が自動起動しないようにする。

[手順]

(1) /etc/inittab ファイルを root 権限で開き、以下の変更をおこなう。

id:5:initdefault　→　id:3:initdefault

(2) /etc/inittab ファイルを保存する。

(3) OS を reboot する。

※ /etc/inittab ファイルは、起動時に読み込まれるファイルである。
修正した id の次の数字は、ランレベルを示しており、3 はマルチユーザ（初期設定）、5 は X-Window の xdm を使った起動モードを、それぞれ表している。

作業 2：メモリ増設対応

[背景] メモリ資源の不足に対処するため実搭載メモリを 48MB から 128MB へ増設した。しかし、BIOS レベルでは 128MB メモリを認識しているにもかかわらず、OS レベルでは 64MB までしか認識していないことが判明した（/var/log/messages の OS ブート時のログより）。情報収集の結果、64MB 以上のメモリを認識するためには、OS の設定が必要であることがわかった。

[手順]

(1) /etc/lilo.conf に、以下の行を付加し、保存する。

append=”mem=128M”

(2) /sbin/lilo コマンドを打ち込み、設定内容を有効にする。

(3) OS を reboot する。

※ 最新の Linux カーネルでは 64MB 以上のメモリでも、自動認識されるようである。

※ /etc/lilo.conf は、マルチブート時の設定などでも編集することがあるが、ここでは、append=”XXXXX”として、カーネルに渡すパラメータラインに指定するパラメータを追加している。自動検出されないハードウェアや自動検出が危険なハードウェアに対するパラメータを指定するのに append は用いられる。

※ Linux では、メインメモリ情報は以下のコマンドによって得られる。

cat /proc/meminfo

作業 3：画面解像度の設定

[背景] 初期インストール時点では、X-Window の画面解像度が 800x600 以上にならなかったため、XFree86 を再インストールして画面解像度の設定をおこなった。

[手順]

(1) XFree86 を再インストールする。

(2) グラフィックの設定をおこなう。

XF86Setup(グラフィカル)、XF86Config(対話型)を使用して以下のように設定した。

グラフィックカード：ATI Mach64 3DRageⅡ

サーバ：SVGA

ディスプレイ(水平周波数)：36.5 - 48.5

(垂直周波数)：50 - 90

解像度：1024×768　　65536 色

作業 4：Samba の設定

[背景] テストセットの準備などにクライアント・サーバ間のファイル共有が有用であるため、Linux 標準添付のファイル共有サーバソフトウェアである Samba の設定をおこなった。

[手順]

(1) /etc/smb.conf に以下の設定を記述する。

```
workgroup = Oita_Workgroup
hosts allow = 192.168.216.0/255.255.255.128
security = share
[public]
comment = Public Stuff
path = /home/samba
public = yes
writable = yes
```

(2) samba デーモンをスタートさせる。

```
# /etc/rc.d/init.d/smb start
```

(3) クライアント端末の識別情報を設定する。

ネットワークコンピュータのプロパティを開き、識別情報のワークグループ名を上記ワークグループ名に設定する。

また、クライアント PC の hosts, lmhosts の両ファイルにサーバ情報を記述する。

3.(4).導入作業

b.リレーショナルデーターベース

i).0racle8i の導入作業

（1）インストール準備

①ルート権限にて Java のランタイム「Java Runtime Emvironment 1.1.6 バージョン５」をインストールする。

```
% cp jre_1.1.6-v5-glibc-x86.tar.gz /usr/local
% cd /usr/local
% tar xvzf jre_1.1.6-v5-glibc-x86.tar.gz
% ln –s /usr/local/jre /usr/local/
```

②ホーム・ディレクトリの作成

```
% mkdir /var/oracle
% mkdir /var/oracle/8i
% mkdir /var/oracle/8i/u01
% mkdir /var/oracle/8i/u02
% mkdir /var/oracle/8i/u03
% mkdir /var/oracle/8i/u04
```

③アカウントの作成

0racle ｲﾝｽﾄｰﾙ・ｸﾞﾙｰﾌﾟ：oinstall

0racle 管理ｸﾞﾙｰﾌﾟ：dba

0racle ｿﾌﾄｳｪｱ所有者：oracle8i(両方のｸﾞﾙｰﾌﾟに属す)

```
% groupadd oinstall
% groupadd dba
% useradd oracle8i –g oinstall –G dba
% passwd oracle8i

% cd /var/oracle
% chown oracle8i.oinstall *
```

④環境変数の設定

以下にファイル「.bash_profile」の内容を示す。

```
# .bash_profile
umask 022
# Get the aliases and functions
#if [ -f ~/.bashrc ]; then
#    . ~/.bashrc
#fi

PATH=$PATH:$HOME/bin:usr/local/bin

# User specific environment and startup programs

ORACLE_BASE=/var/oracle/8i/u01/app/oracle
ORACLE_HOME=$ORACLE_BASE/product/8.1.5
export ORACLE_BASE ORACLE_HOME
```

```
NLS_LANG='japanese_japan.ja16euc'
ORA_NLS33=$ORACLE_HOME/ocommon/nls/admin/data
LD_LIBLARY_PATH=$LD_LIBLARY_PATH:$ORACLE_HOME/lib


PATH=$PATH:$ORACLE_HOME/bin
export NLS_LANG ORA_NLS33 LD_LIBLARY_PATH PATH

JAVA_HOME=/usr/local/jre
export JAVA_HOME

PATH=$JAVA_HOME/bin:$PATH

ORACLE_SID=ORCL
export ORACLE_SID

#BASH_ENV=$HOME/.bashrc
#USERNAME=""

#export USERNAME BASH_ENV
```

.bash_profile に設定する環境変数を有効にするため、以下のコマンドを実行する。

```
% source .bash_profile
```

（2）インストール実行

① インストーラの起動

・Oracle8i の CD-ROM 挿入

```
% su oracle8i
% cd /mnt/cdrom
% ./runInstaller
```

②ソースおよびインストール先の指定

「Source」のパス：/mnt/cdrom/stage/products.jar

「Distination」のパス：/var/oracle/8i/u01/app/oracle/product/8.1.5

③UNIX グループの指定

「oinstall」

④orainstRoot.sh の実行

```
% su
# /tmp/OraInstall/orainstRoot.sh
```

⑤インストール製品の選択

「Oracle8i 8.1.5.0.0」

⑥インストールの種類の選択

「Custom」

⑦コンポーネントの選択

「Oracle interMedia 8.1.5.0.0」のチェックを外します。

⑧言語の選択

「Japanese」

⑨データベース作成の選択

Yes

⑩データベース名と SID の設定

グローバル・データベース名：「oisa」

SID：「ORCL」

⑪DB ファイルの格納場所の指定

/var/oracle/8i/u02

⑫インストールの開始

⑬root.sh の実行

/var/oracle/8i/u01/app/oracle/product/8.1.5/root.sh

「local bin のパス」：/usr/local/bin

（3）データベースの作成

①データベース・タイプの選択

「カスタム」

②動作環境の設定

「ハイブリッド」

③同時接続ユーザ数の指定

「5」

④稼動モードの選択

「専用サーバー・モード」

⑤カートリッジとオプションの選択

「Oracle JServer」、「アドバンスド - レプリケーション」、

「カートリッジのデータを使用可能にし、SQL*Plus ヘルプテーブルを…」

⑥データベース名と SID の設定

グローバル・データベース名：「oisa」

SID：「ORCL」

初期化パラメータファイル：

「/var/oracle/8i/u01/app/oracle/admin/oisa/pfile/initoracle.ora」

⑦キャラクタセットの設定

「JA16EUC」

⑧制御ファイル指定

「/ var/oracle/8i/u01/app/oracle/oradata/oisa/control01.ctl」

「/ var/oracle/8i/u01/app/oracle/oradata/oisa/control02.ctl」

⑨各種表領域の設定

「/var/oracle/8i/u02/oradata/oisa/system01.dbf」

「/var/oracle/8i/u02/oradata/oisa/user01.dbf」

⑩REDO ログの指定

「/var/oracle/8i/u03/oradata/oisa/redo01.log」

「/var/oracle/8i/u03/oradata/oisa/redo02.log」

⑪ロギング・パラメータ情報の設定

アーカイブ・ログを使用不可にする。

⑫SGA のパラメータ情報の設定

共有プールサイズ：「52428800 バイト」

ブロック・バッファ数：「8192」

ログバッファサイズ：「163840 バイト」

プロセス数：「50」

ブロックサイズ：「2048 バイト」

⑬DUMP ファイルの指定

トレースファイルのディレクトリ：

「background_dump_dest = /var/oracle/8i/u01/app/oracle/admin/yubin_db/bdump」

「core_dump_dest = /var/oracle/8i/u01/app/oracle/admin/yubin_db/cdump」

「user_dump_dest = /var/oracle/8i/u01/app/oracle/admin/yubin_db/udump」

⑭データベースの作成開始

「データベースの即時作成」

⑮作成完了

メモリ 128M で約３時間要する。

（４）インストール後の確認

①プロセスの実行確認

```
% ps ax | grep ora
```

②データベースの動作確認

```
% svrmgrl
SVRMGRL> connect system/manager
SVRMGRL> @?/rdbms/admin/utlsampl.sql
SVRMGRL> connect scott/tiger
SVRMGRL> select * from emp;
```

（５）データベースの設定

①リスナーの起動

vi /var/oracle/8i/u01/app/oracle/product/8.1.5/network/admin/listener.ora

HOST 名、ORACLE_HOME、グローバル・データベース名、SID などを修正

```
$ lsnrctl
LSNRCTL> start
```

②ロールバックセグメント

```
$ svrmgrl
SVRMGRL> connect internal
SVRMGRL> shutdown
```

初期化パラメータファイル

/var/oracle/8i/u01/app/oracle/admin/oisa/pfile/initoracle.ora を編集する。

rollback_segments = (r01, r02, r03, r04)　←　コメントをはずす

```
SVRMGRL> connect internal
SVRMGRL> startup
```

以下にファイル「initoracle.ora」の内容を示す。

```
db_name = oisa
instance_name = ORCL

service_names = oisa


control_files = ("/var/oracle/8i/u01/app/oracle/oradata/oisa/control01.ctl",
"/var/oracle/8i/u01/app/oracle/oradata/oisa/control02.ctl")


db_block_buffers = 8192

shared_pool_size = 15728640

java_pool_size = 20971520

log_checkpoint_interval = 10000
log_checkpoint_timeout = 1800

processes = 50

log_buffer = 163840

# audit_trail = false  # if you want auditing
# timed_statistics = false  # if you want timed statistics
# max_dump_file_size = 10000  # limit trace file size to 5M each

# If using private rollback segments, place lines of the following
# form in each of your instance-specific init.ora files:
 rollback_segments = (r01, r02, r03, r04)

# Global Naming -- enforce that a dblink has same name as the db it connects to
# global_names = false

# Uncomment the following line if you wish to enable the Oracle Trace product
# to trace server activity.  This enables scheduling of server collections
# from the Oracle Enterprise Manager Console.
# Also, if the oracle_trace_collection_name parameter is non-null,
# every session will write to the named collection, as well as enabling you
# to schedule future collections from the console.
# oracle_trace_enable = true

# define directories to store trace and alert files
background_dump_dest = /var/oracle/8i/u01/app/oracle/admin/oisa/bdump
core_dump_dest = /var/oracle/8i/u01/app/oracle/admin/oisa/cdump
user_dump_dest = /var/oracle/8i/u01/app/oracle/admin/oisa/udump

db_block_size = 2048
```

```
remote_login_passwordfile = exclusive

os_authent_prefix = ""

# The following parameters are needed for the Advanced Replication Option
job_queue_processes = 0
job_queue_interval = 60
distributed_transactions = 10
open_links = 4

mts_dispatchers = "(PROTOCOL=TCP)(PRE=oracle.aurora.server.SGiopServer)"
# Uncomment the following line when your listener is configured for SSL
# (listener.ora and sqlnet.ora)
# mts_dispatchers = "(PROTOCOL=TCPS)(PRE=oracle.aurora.server.SGiopServer)"

mts_servers = 1
compatible = "8.1.0"
```

（６）自動起動の設定

今回は、サービス起動方法等の手法を勉強する事とした為、サービスは手動で起動した。
これらはの起動は、ＯＳの環境ファイルに記述することで、自動起動が可能である。

ii).PostgreSQL の導入作業

（1）データベース（PostgreSQL）のセットアップ

注：PostgreSQL のインストールは、RedHatLinux のインストール時に、行なわれている。

①データベースの初期化

```
# initdb
```

②デーモンプロセス（postmaster）を起動した。

```
# /etc/rc.d/init.d/postgresql start
# /etc/init.d/postgresql status
```

③PostgreSQL 管理者のパスワード設定

```
# passwd postgres
```

「postgres のパスワード：oisa-linux」

```
# su – postgres
```

④データベースの作成

```
$ createdb sampledb
```

※データベースの実体ファイルは“ /usr/lib/pgsql/base ”ディレクトリ配下に作成される。

⑤認証ファイルの設定

[対処] ファイル“ /var/lib/pgsql/pg_hba.conf ”に以下の行を追加する。

```
host all 0.0.0.0 0.0.0.0 trust
```

※DB やサーバのの再起動は必要ない。

⑥サーバ起動時にデーモンプロセス（postmaster)も自動的に起動するように設定した。

```
# /sbin/chkconfig –add postgresql
```

（2）データベース設定後の確認

①psql コマンドを用いて、フロントエンドで、クエリーが発行できる事を確認した。

```
$ psql sampledb
sampledb => select * from emp;
```

②クライアントからの ODBC 経由での接続確認

・クライアント PC に以下の場所から入手した ODBC ドライバをインストールした。

http://www.interwiz.koganei.tokyo.jp/software/PsqlODBC/index.html

（注）ODBC ドライバの設定で“ Read Only ”のチェックマークをはずす。

（3）可視化ツール（PgAccess）のインストール

①pgaccess のインストール

・RedhatLinux 初期インストール時で、カスタムインストールを選択しないと、pgaccess はインストールされないことを確認。

・pgaccess は、postgresql-tcl-6.5.3-6.i386.rpm パッケージからインストールできる。

```
rpm –i postgresql-tcl-6.5.3-6.i386.rpm
```

ここでは、依存関係によるエラー（libncurses.so が存在しない）が発生。

・ncurses に関する rpm パッケージを全てインストール

```
rpm –i ncurses-devel-5.0-11.i386.rpm
rpm –i ncurses3-1.9.9e-11.i386.rpm
```

rpm –i ncurses-5.0-11.i386.rpm

※ rpm –q –l ncurses-5.0-11 とすると、RPM パッケージ内容が見ることができる。

・postgre_sql_tcl.i386.rpm パッケージの再インストール

rpm –i postgresql-tcl-6.5.3-6.i386.rpm

・root 権限で、/usr/bin/pgaccess を起動

②テーブルのエクスポートの試用

テーブルのエクスポートを行なうと、postgres のデータ格納ディレクトリ（/var/lib/pgsql/base/データベース名/テーブル名.txt）に格納される。

４.(1).動作検証及び評価（仮想クライアントでのベンチマークテスト）【準備段階】

Postgresql と Oracle にて仮想クライアントでのベンチマークテストを実施。
実施するにあたり、MySQL 開発者作成による Java プログラム「JDBCBench」をもとに今回評価用にカスタマイズしたものを利用した。そもそもこのプログラムは銀行口座トランザクションをモデル化したベンチマーク仕様「TPC-B」に基づいて設計されたものである。

「JDBCBench」(http://www.worldserver.com/mm.mysql/performance/)

（１）仮想クライアントでのベンチマークテスト用 Java プログラムのカスタマイズ

「JDBCBench」を Windows クライアント端末から Linux サーバにアクセスするため、以下の変更を加えている。

①ユーザ ID、パスワードの起動パラメータ化
②Statements オブジェクトの明示的なクローズ
③Oracle8i 版では、timestamp 型の代わりに、date 型を使用。

（２）実行環境

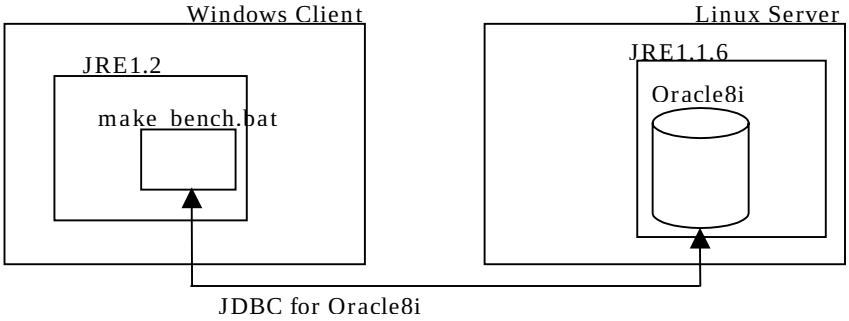


図４.(1).－１　ベンチマーク実行環境（Oracle8i）

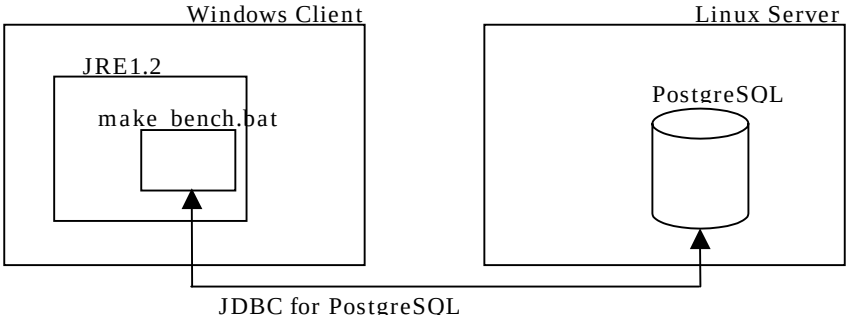


図４.(1).－２　ベンチマーク実行環境（PostgreSQL）

α）Windows Client に用意する実行環境（oracle8i-java フォルダ）

make_bench.bat：仮想クライアントでのベンチマークテスト用実行ファイル

※-init は、テーブルを生成してからベンチマークを行う。ベンチマークの１回目だけこの-init をつけ、２回目以降は、-init を外す。

※-tpc n は、クライアントあたりに発行するトランザクション数で、シミュレーションプログラムへの入力パラメータである。

※-clients　n は、接続クライアント数で、シミュレーションプログラムへの入力パラメータである。

※-user XXXX は、接続ユーザ名を指定する。（linux）

※-pass xxxx は、接続ユーザのパスワードを指定する。（linux）

classes12.zip：Oracle8i 用 JDBC ドライバ

β）Windows Client に用意する実行環境（postgresql-java フォルダ）

make_bench.bat：仮想クライアントでのベンチマークテスト用実行ファイル

※-init は、テーブルを生成してからベンチマークを行う。ベンチマークの１回目だけこの-init をつけ、２回目以降は、-init を外す。

※-tpc n は、クライアントあたりに発行するトランザクション数で、シミュレーションプログラムへの入力パラメータである。

※-clients n は、接続クライアント数で、シミュレーションプログラムへの入力パラメータである。

※-user XXXX は、接続ユーザ名を指定する。（linux）

※-pass xxxx は、接続ユーザのパスワードを指定する。（linux）

postgresql.jar：PostgreSQL 用 JDBC ドライバ

（注）以下のベンチマークを行なう際には、X、SAMBA など不要プロセスの停止する。

（３）ベンチマークテストの入出力パラメータ

α）入力パラメータ（可変）

①１クライアントあたりに発行できるトランザクション数

※make_bench.bat の-tpc で指定

1,10,20,100

②接続クライアント数

※make_bench.bat の-clients で指定

1,5,10,20

β）出力パラメータ

①Client における最大使用メモリ（KB）

②Client における最小使用メモリ（KB）

③総トランザクション数の発行に要した時間（秒）

④単位時間あたりのトランザクション数（回数／秒）

⑤総トランザクション数のうち発行に失敗したトランザクション数（回）

（参考１）仮想クライアントでのベンチマークテストで用いるテーブル一覧

（１）branches（支店テーブル）　※支店数：１

No	フィールド名	属性	キー	索引	備考
1	Bid	**int**	○		0,1,2,3,･･･,支店数-1：支店コード
2	**Bbalance**	int			“0”
3	**filler**	**char(84)**			d.c.

（２）tellers（窓口テーブル）　※窓口数：１０

No	フィールド名	属性	キー	索引	備考
1	Tid	**int**	○		0,1,2,3,･･･,窓口数-1：窓口コード
2	**Bid**	int			0,0,0,･･･････　支店コード
3	**Tbalance**	**int**			“0”
4	**filler**	**char(84)**			d.c.

（３）accounts（明細テーブル）　※明細数：１０００００

No	フィールド名	属性	キー	索引	備考
1	Aid	**int**	○		0,1,2,3,･･･,明細数-1：明細コード
2	**Bid**	int			0,0,0,･･･････　支店コード
3	**Abalance**	**int**			“0”
4	**filler**	**char(84)**			d.c.

（４）history（履歴テーブル）　※履歴数：０

No	フィールド名	属性	キー	索引	備考
1	Tid	**int**			窓口コード
1	Bid	**int**			支店コード
2	**Aid**	int			明細コード
3	**delta**	**int**			0～1000 までの間の乱数
4	**time**	**timestamp**			更新時間
5	**filler**	**char(22)**			d.c.

（参考２）仮想クライアントでのベンチマークテスト用 Java プログラム（PostgreSQL 版）

別添１．１．：『仮想クライアントでのベンチマークテスト用 Java プログラム（PostgreSQL 版）』参照

（参考３）仮想クライアントでのベンチマークテスト用 Java プログラム（Oracle8i 版）

別添１．２．：『仮想クライアントでのベンチマークテスト用 Java プログラム（Oracle8i 版）』参照

（参考４）make_bench.bat（PostgreSQL 版）

```
cd C:\WINDOWS\ﾃﾞｽｸﾄｯﾌﾟ\Url_0110\benchmark\postgresql-java

rem javac -classpath $CLASSPATH;postgresql.jar JDBCBench.java

set CLASSPATH=

rem java -classpath .;postgresql.jar JDBCBench -driver postgresql.Driver -url jdbc:postgresql://athena/yubin_db#EUCJIS -tpc 10 -clients 10  -init -user pg_linux -pass pg_linux
C:\Progra~1\JavaSoft\JRE\1.2\bin\java -classpath .;postgresql.jar JDBCBench -driver postgresql.Driver -url jdbc:postgresql://192.168.1.1/yubin_db#EUCJIS -tpc 10 -clients 50 -v -user pg_linux -pass pg_linux
```

（参考５）make_bench.bat（Oracle8i 版）

```
cd C:\WINDOWS\ﾃﾞｽｸﾄｯﾌﾟ\Url_0110\benchmark\oracle8i-java

set CLASSPATH=
rem javac -classpath .;classes12.zip JDBCBench.java
rem java -classpath .;classes12.zip JDBCBench  -driver oracle.jdbc.driver.OracleDriver -url jdbc:oracle:thin:@athena:1521:oracle -tpc 10 -clients 10 -init -user linux -pass linux
C:\Progra~1\JavaSoft\JRE\1.2\bin\java -classpath .;classes12.zip JDBCBench -driver oracle.jdbc.driver.OracleDriver -url jdbc:oracle:thin:@192.168.1.1:1521:oracle -tpc 10 -clients 50 -v -user linux -pass linux
```

（参考６）JRE１．２

ベンチマークテストを行うには、最新の Java 実行環境（JRE1.2）をインストールしておく必要がある。

４.(2).動作検証及び評価（実クライアントでのベンチマークテスト）【準備段階】

Postgresql と 0racle にてベンチマークテストを実施。
実施するにあたり、簡単なクエリーが発行できる Java プログラムを作成した。
データには、郵政省が無償で配布している郵便番号データ（約１２万件）を用いた。

a.検証用プログラムの準備

（１）実クライアントでのベンチマークテスト用 Java プログラムの作成

①ユーザ ID、パスワード、URL、発行するクエリーの設定ファイル化（sg.txt）
②0racle8i 版では、文字列を varchar2 で、PostgreSQL 版では、文字列を varchar で表した。

（２）実行環境

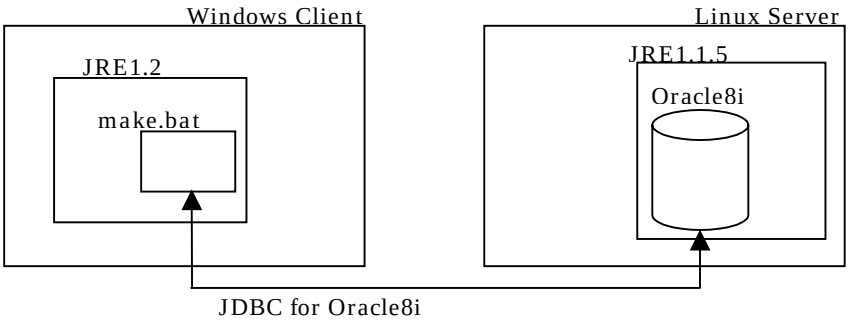


図４.(2).－１　ベンチマーク実行環境（Oracle8i）

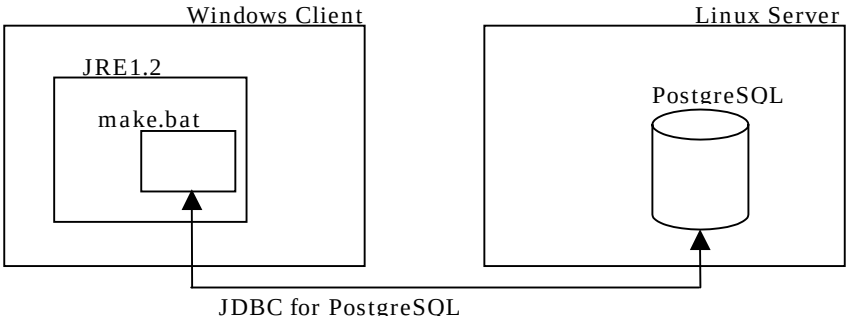


図４.(2).－２　ベンチマーク実行環境（PostgreSQL）

α）Windows Client に用意する実行環境（oracle8i-java フォルダ）

make.bat：実クライアントでのベンチマークテスト用実行ファイル
sg.txt：make.bat で用いる環境設定ファイル
※デリミタは、セミコロンで、以下の順に記述する。
①サーバ IP
②SID 名(ORCL)
③接続ユーザ名(linux)
④接続ユーザのパスワード(linux)
⑤発行するクエリー文
classes12.zip：Oracle8i 用 JDBC ドライバ

β）Windows Client に用意する実行環境（postgresql-java フォルダ）

make.bat：実クライアントでのベンチマークテスト用実行ファイル

sg.txt：make.bat で用いる環境設定ファイル

※デリミタは、セミコロンで、以下の順に記述する。

①サーバ IP

②データベース名(oisadb)

③接続ユーザ名(linux)

④接続ユーザのパスワード(linux)

⑤発行するクエリー文

postgresql.jar：PostgreSQL 用 JDBC ドライバ

（注）以下のベンチマークを行なう際には、X、SAMBA など不必要なプロセスは停止する。

（3）ベンチマークテストの入出力パラメータ

α）入力パラメータ（可変）：

①sg.txt ファイルのクエリー文

②接続するクライアント数（1台）

β）出力パラメータ：

①接続時間

②クエリー発行時間

③接続時間＋クエリー発行時間

b.サーバ環境

（1） Oracle8i

①DB 生成、SID 設定

% dbassist

（SID：ORCL、DB 名：oisa）

②ユーザ生成

% svrmgrl

SVRMGRL> connect internal

SVRMGRL> startup

% sqlplus (system/manager でログイン)

＞@/tmp/oracle/create_user.sql

以下に、「create_user.sql」の内容を示す.

```
create user linux identified by linux
```

⇒ユーザ linux をパスワード linux で生成。

※ デフォルトでは、クライアント端末から接続するためのリスナーが動いていないため、設定ファイルを修正して、リスナーを起動してやる必要がある。

%cd $ORACLE_HOME/network/admin

%cp ./samples/listner.ora .

listner.ora の LISTNER 句、SID_LIST_LISTNER 句を修正。

⇒サーバ名、グローバル DB 名、SID 名、環境変数 ORACLE_HOME のパスなど
を修正。

%lsnrctl

リスナーを起動します。

LSNRCTL>start

（２） PostgreSQL

①DB 生成

% createdb oisadb

⇒時間がかかるため、DB は、サンプル環境のものを使用。（DB 名：oisadb）

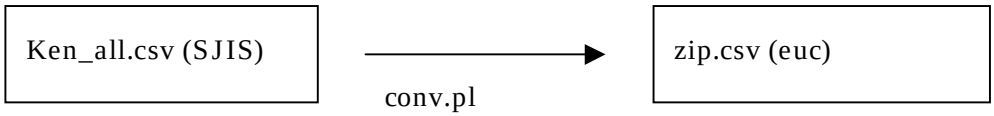
②ユーザ生成

% createuser linux password with linux

⇒　user's postgres ID を 56 に設定。

c.テーブル生成・データロード・インデックス生成

（１）データロードの準備



郵便番号（全国）
120253 レコード

郵便番号（全国）

SJIS→EUC

①郵便データは、ダウンロードした段階では、文字コードがＳＪＩＳ、改行コードは、
ＭＳ－ＤＯＳの CR+LF、半角かな混じりである（Ken_all.csv）。

②perl スクリプト（conv.pl）を用いて、半角かなは全角かなへ、文字コードは、EUC へ変換する。

※このスクリプトは、jcode.pl が必要なので、カレントにコピーしておく。

以下に conv.pl の内容を示す。

```
#!/usr/bin/perl
require 'jcode.pl';
while (<>){ s/\r\n$/\n/; # delete CR
&jcode'h2z_sjis(*_); # 1byte kana to 2byte kana
&jcode'sjis2euc(*_); # shift JIS to EUC
s/"//g; # delete double quote
## s/,/\t/g; # , to tab
print; # output
}
```

ここで、変換された zip.csv は、^Z という EOF が入っているので、vi で開いて、
手動で、取り除く。

① vi zip.csv
② :120253 で、指定行に移動。
③ 行末の^Z というところにカーソルを移動し、dd とコマンド入力し、
行末の^Z を取り除く。
④ wc –l zip.csv とし、120253 となれば、OK.

oisa_benchmark_pre.tar.Z をサーバにおとす。

```
cp /home/samba/ oisa_benchmark_pre.tar.Z /tmp
cd /tmp
uncompress oisa_benchmark_pre.tar.Z
tar xvf oisa_benchmark_pre.tar
```

/tmp/oisa_benchmark_pre というディレクトリの直下に、先ほど作成した
zip.csv をコピーする。

（2）テーブル生成（2分）

α）Oracle

```
cd /tmp/oisa_benchmark_pre/oracle
sqlplus linux/linux
>  @create_table_oracle.sql
>  quit
```

以下にファイル「create_table_oracle.sql」の内容を示す。

```
create table yubin_table1
(
yubin_no        varchar2(7),
kanji_ken       varchar2(8),
kanji_si        varchar2(24),
kanji_cho       varchar2(74)
);

create table yubin_table2
(
yubin_no        varchar2(7),
kana_ken       varchar2(16),
kana_si        varchar2(48),
kana_cho       varchar2(148)
);

create table yubin_table3
(
yubin_no        varchar2(7),
kanji_ken       varchar2(8),
kanji_si        varchar2(24),
kanji_cho       varchar2(74),
```

```
kana_ken        varchar2(16),
kana_si         varchar2(48),
kana_cho        varchar2(148),
cho2_flg varchar2(1),
chome_flg         varchar2(1),
yubin2_flg        varchar2(1)
);
```

β）PostgreSQL

cd /tmp/oisa_benchmark_pre/postgres

psql oisadb

> ¥i create_table_postgres.sql

> ¥q

以下にファイル「create_table_postgres.sql」の内容を示す。

```
create table yubin_table1
(
yubin_no        varchar(7),
kanji_ken       varchar(8),
kanji_si        varchar(24),
kanji_cho       varchar(74)
);

create table yubin_table2
(
yubin_no        varchar(7),
kana_ken       varchar(16),
kana_si        varchar(48),
kana_cho       varchar(148)
);

create table yubin_table3
(
yubin_no        varchar(7),
kanji_ken       varchar(8),
kanji_si        varchar(24),
kanji_cho       varchar(74),
kana_ken       varchar(16),
kana_si        varchar(48),
kana_cho       varchar(148),
cho2_flg varchar(1),
chome_flg         varchar(1),
yubin2_flg        varchar(1)
);
```

（3）データロードの準備（2分）

α）Oracle8i

./awk_oracle_loader_files.sh

以下にファイル「awk_oracle_loader_files.sh」の内容を示す。

```
#!/bin/sh

#./conv.pl KEN_ALL.CSV > zip.csv
awk -F, ' { print $3 "," $7 "," $8 "," $9 } ' ../zip.csv > yubin_oracle1.csv

awk -F, ' { print $3 "," $4 "," $5 "," $6 } ' ../zip.csv > yubin_oracle2.csv

awk -F, ' { print $3 ",oisa,oisa,oisa,oisa,oisa,oisa," $10 "," $12 "," $13 } ' ../zip.csv
> yubin_oracle3.csv
```

β）PostgreSQL

./awk_postgre_loader_files.sh

以下にファイル「awk_postgre_loader_files.sh」の内容を示す。

```
#!/bin/sh

#./conv.pl KEN_ALL.CSV > zip.csv
awk -F, ' { print $3 " " $7 "  " $8 "  " $9 } ' ../zip.csv > yubin_postgres1.csv

awk -F, ' { print $3 " " $4 "  " $5 "  " $6 } ' ../zip.csv > yubin_postgres2.csv

awk -F, ' { print $3 " oisa    oisa    oisa    oisa    oisa    oisa    " $10 " "
$12 " " $13 } ' ../zip.csv > yubin_postgres3.csv
```

（4）データロード（１５分）

α）Oracle8i

./oracle_loader_kanji.sh

./oracle_loader_kana.sh

./oracle_loader_all.sh

以下に、ファイル「oracle_loader_kanji.sh」の内容を示す。

```
#!/bin/sh  -x
time sqlldr linux/linux yubin_ldr_oracle1.ctl >& /tmp/benchmark_pre_oracle001
```

以下に、ファイル「yubin_ldr_oracle1.ctl」の内容を示す。

```
load data
infile 'yubin_oracle1.csv'
append into table yubin_table1
fields terminated by ","
(
yubin_no,
kanji_ken,
kanji_si,
kanji_cho
)
```

以下に、ファイル「oracle_loader_kana.sh」の内容を示す。

```
#!/bin/sh  -x

time sqlldr linux/linux yubin_ldr_oracle2.ctl >& /tmp/benchmark_pre_oracle002
```

以下に、ファイル「yubin_ldr_oracle2.ctl」の内容を示す。

```
load data
infile 'yubin_oracle2.csv'
append into table yubin_table2
fields terminated by ","
(
yubin_no,
kana_ken,
kana_si,
kana_cho
)
```

以下に、ファイル「oracle_loader_all.sh」の内容を示す。

```
#!/bin/sh  -x

time sqlldr linux/linux yubin_ldr_oracle3.ctl >& /tmp/benchmark_pre_oracle003
```

以下に、ファイル「yubin_ldr_oracle3.ctl」の内容を示す。

```
load data
infile 'yubin_oracle3.csv'
append into table yubin_table3
fields terminated by ","
(
yubin_no,
kanji_ken,
kanji_si,
kanji_cho,
kana_ken,
kana_si,
kana_cho,
cho2_flg ,
chome_flg,
yubin2_flg
)
```

β）PostgreSQL

./postgre_loader_kanji.sh

./postgre_loader_kana.sh

./postgre_loader_all.sh

以下に、ファイル「postgre_loader_kanji.sh」の内容を示す。

```
#!/bin/sh

echo " \copy yubin_table1 from yubin_postgres1.csv " | time psql oisadb >&
/tmp/benchmark_pre_postgres001
```

以下に、ファイル「postgre_loader_kana.sh」の内容を示す。

```
#!/bin/sh

echo " \copy yubin_table2 from yubin_postgres2.csv " | time psql oisadb >&
/tmp/benchmark_pre_postgres002
```

以下に、ファイル「postgre_loader_all.sh」の内容を示す。

```
#!/bin/sh

echo " \copy yubin_table3 from yubin_postgres3.csv " | time psql oisadb >&
/tmp/benchmark_pre_postgres003
```

（５）インデックス作成（２０分）

α）Oracle8i

sqlplus linux/linux

> @create_index_common.sql

> quit

β）PostgreSQL

psql oisadb

> ¥i create_index_common.sql

> ¥q

以下にファイル「create_index_common.sql」の内容を示す。

```
create index yubin_no_ndx1 on yubin_table1(yubin_no);
create index kanji_ken_ndx1 on yubin_table1(kanji_ken);
create index kanji_si_ndx1 on yubin_table1(kanji_si);
create index kanji_cho_ndx1 on yubin_table1(kanji_cho);

create index yubin_no_ndx2 on yubin_table2(yubin_no);
create index kana_ken_ndx2 on yubin_table2(kana_ken);
create index kana_si_ndx2 on yubin_table2(kana_si);
create index kana_cho_ndx2 on yubin_table2(kana_cho);

create index yubin_no_ndx3 on yubin_table3(yubin_no);
create index kanji_ken_ndx3 on yubin_table3(kanji_ken);
create index kanji_si_ndx3 on yubin_table3(kanji_si);
create index kanji_cho_ndx3 on yubin_table3(kanji_cho);
create index kana_ken_ndx3 on yubin_table3(kana_ken);
create index kana_si_ndx3 on yubin_table3(kana_si);
create index kana_cho_ndx3 on yubin_table3(kana_cho);
```

（参考１）実クライアントでのベンチマークテスト用 Java プログラム（PostgreSQL 版・参照系）

別添１．３．:『実クライアントでのベンチマークテスト用Javaプログラム（PostgreSQL版・参照系）』参照

（参考２）実クライアントでのベンチマークテスト用 Java プログラム（PostgreSQL 版・更新系）

別添１．４．:『実クライアントでのベンチマークテスト用Javaプログラム（PostgreSQL版・更新系）』参照

（参考３）実クライアントでのベンチマークテスト用 Java プログラム（Oracle8i 版・参照系）

別添１．５．:『実クライアントでのベンチマークテスト用Javaプログラム（Oracle8i版・参照系）』参照

（参考４）実クライアントでのベンチマークテスト用 Java プログラム（Oracle8i 版・更新系）

別添１．６．:『実クライアントでのベンチマークテスト用Javaプログラム（Oracle8i版・更新系）』参照

（参考５）ベンチマークテストで用いる SQL 文一覧

評価項目として、表４.(2).5.－１の SQL 文を発行し、各処理時間のデータ収集を１度ずつ行う。

表４.(2).５－１　使用 SQL 文一覧

	SQL 文
文字列完全一致	Select * from yubin_table1 where yubin_no = '1300014' ;
文字列前方一致	Select * from yubin_table1 where kanji_cho like '新町%' ;
文字列部分一致	Select * from yubin_table1 where kanji_cho like '%新町%' ;
AND 条件検索	Select * from yubin_table1 where kanji_ken like '%京%' and kanji_si like '%大%' and kanji_cho like '%本町%' ;
全件挿入	insert into yubin_table1_cp (select * from yubin_table1 where kanji_ken like '大分県%' ;
全件更新	update yubin_table1_cp set kanji_ken = 'oisa' ;
全件削除	delete from yubin_table1_cp ;
ﾃｰﾌﾞﾙ結合	select a.yubin_no , b.kanji_ken from yubin_table1 b , yubin_table3 a where a.yubin2_flg = '0' and a.yubin_no = b.yubin_no ;

（参考６）ベンチマークテストでテーブル構成（PostgreSQL 版）

（１）yubin_table1

No	フィールド名	属性	キー	索引	備考
1	**Yubin_no**	**Varchar(7)**		○	郵便番号
2	**Kanji_ken**	**Varchar(8)**		○	都道府県名（漢字）
3	**Kanji_si**	**Varchar(24)**		○	市町村区名（漢字）
4	**Kanji_cho**	**Varchar(74)**		○	町域以下（漢字）

（２）yubin_table2

No	フィールド名	属性	キー	索引	備考
1	**yubin_no**	**Varchar(7)**		○	郵便番号
2	**kana_ken**	**Varchar(16)**		○	都道府県名（カナ）
3	**kana_si**	**Varchar(48)**		○	市町村区名（カナ）
4	**kana_cho**	**Varchar(148)**		○	町域以下（カナ）

（3）yubin_table3

No	フィールド名	属性	キー	索引	備考
1	**yubin_no**	**Varchar(7)**		○	郵便番号
2	**kanji_ken**	**Varchar(8)**		○	都道府県名（漢字）　初期値 “oisa”
3	**kanji_si**	**Varchar(24)**		○	市町村区名（漢字）　初期値 “oisa”
4	**kanji_cho**	**Varchar(74)**		○	町域以下（漢字）　初期値 “oisa”
5	**kana_ken**	**Varchar(16)**		○	都道府県名（カナ）　初期値 “oisa”
6	**Kana_si**	**Varchar(48)**		○	市町村区名（カナ）　初期値 “oisa”
7	**Kana_cho**	**Varchar(148)**		○	町域以下（カナ）　初期値 “oisa”
8	**cho2_flg**	**Varchar(1)**			町域重複フラグ
9	**chome_flg**	**Varchar(1)**			丁目フラグ
10	**yubin2_flg**	**Varchar(1)**			郵便番号重複フラグ

※町域重複フラグ：一町域が二以上の新郵便番号で表される場合の表示　(「1」は該当、「0」は該当せず)

※丁目フラグ：丁目を有する町域の場合の表示　(「1」は該当、「0」は該当せず)

※郵便番号重複フラグ：一つの新郵便番号で二以上の町域を表す場合の表示　(「1」は該当、「0」は該当せず)

（参考７）ベンチマークテストでテーブル構成（0racle8i 版）

（1）yubin_table1

No	フィールド名	属性	キー	索引	備考
1	**Yubin_no**	**varchar2(7)**		○	郵便番号
2	**Kanji_ken**	**varchar2(8)**		○	都道府県名（漢字）
3	**Kanji_si**	**varchar2(24)**		○	市町村区名（漢字）
4	**Kanji_cho**	**varchar2(74)**		○	町域以下（漢字）

（2）yubin_table2

N	フィールド名	属性	キー	索引	備考

o					
1	Yubin_no	varchar2(7)		○	郵便番号
2	kana_ken	varchar2(16)		○	都道府県名（カナ）
3	kana_si	varchar2(48)		○	市町村区名（カナ）
4	kana_cho	varchar2(148)		○	町域以下（カナ）

（3）yubin_table3

No	フィールド名	属性	キー	索引	備考
1	Yubin_no	varchar2(7)		○	郵便番号
2	Kanji_ken	varchar2(8)		○	都道府県名（漢字）　初期値 “oisa”
3	Kanji_si	varchar2(24)		○	市町村区名（漢字）　初期値 “oisa”
4	Kanji_cho	varchar2(74)		○	町域以下（漢字）　初期値 “oisa”
5	kana_ken	varchar2(16)		○	都道府県名（カナ）　初期値 “oisa”
6	kana_si	varchar2(48)		○	市町村区名（カナ）　初期値 “oisa”
7	kana_cho	varchar2(148)		○	町域以下（カナ）　初期値 “oisa”
8	cho2_flg	varchar2(1)			町域重複フラグ
9	Chome_flg	varchar2(1)			丁目フラグ
10	Yubin2_flg	varchar2(1)			郵便番号重複フラグ

※町域重複フラグ：一町域が二以上の新郵便番号で表される場合の表示　(「1」は該当、「0」は該当せず)

※丁目フラグ：丁目を有する町域の場合の表示　(「1」は該当、「0」は該当せず)

※郵便番号重複フラグ：一つの新郵便番号で二以上の町域を表す場合の表示　(「1」は該当、「0」は該当せず)

（参考８）make.bat（PostgreSQL 版）

```
cd C:¥WINDOWS¥ﾃﾞｽｸﾄｯﾌﾟ¥postgresql-java
javac -classpath $CLASSPATH;postgresql.jar test.java
set CLASSPATH=
java -classpath .;postgresql.jar test
```

（参考９）sg.txt（PostgreSQL 版）

```
athena ; yubin_db ; pg_linux ; pg_linux ; SELECT * FROM yubin_table3 where yubin_no like '%2800%' ;
```

（参考１０）make.bat（Oracle8i 版）

```
cd C:¥WINDOWS¥ﾃﾞｽｸﾄｯﾌﾟ¥oracle8i-java
set CLASSPATH=
```

```
javac -classpath .;classes12.zip test.java
java -classpath .;classes12.zip test
```

（参考１１）sg.txt（Oracle8i 版）

```
athena ; oracle ; linux ; linux ; SELECT * FROM yubin_table3 where yubin_no like
'%2800%' ;
```

４.(3).評価分析

ａ．仮想クライアントでのベンチマークテスト

以下の表のように、仮想クライアント数および１クライアントあたりに発行できるトランザクション数を変動させ、１秒間で処理できるトランザクション数およびそれにかかわったメモリ使用量のデータ収集を１度ずつ行った。

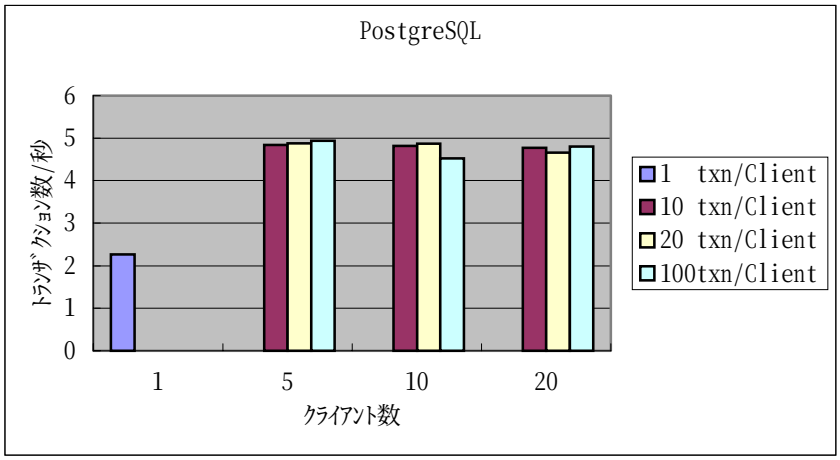
＜１秒間に処理可能なトランザクション数＞　(単位：txn/sec)

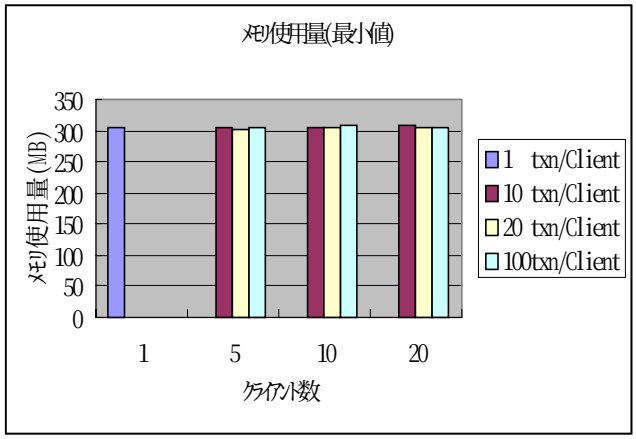
tpc / Client	1	10	20	100
1	2.27	—	—	—
5	—	4.84	4.88	4.94
10	—	4.82	4.87	4.52
20	—	4.77	4.60	4.80

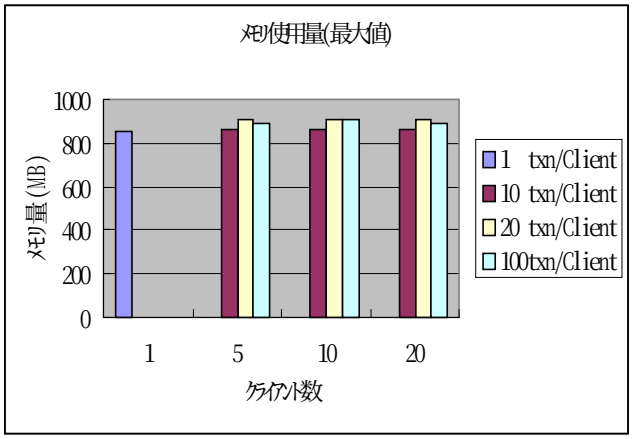
＜使用メモリ最大／最小量＞　(単位：mb)

tpc / Client	1	10	20	100
1	852/306	—	—	—
5	—	861/306	863/306	861/307
10	—	912/302	912/306	912/306
20	—	886/304	906/308	886/306

i.)PostgreSQL の評価分析







データ収集結果をもとに、上図のようベンチマーク結果をグラフに表すことにより、PostgreSQL ではクライアント数および１クライアントあたりの発行トランザクション数の変動に関わらず１秒間あたり約５トランザクションを処理していることが分かる。
また、実際の検証時には常にハードディスクへアクセスが発生しており、このことからも PostgreSQL では、高性能ＣＰＵおよび回転数の速いハードディスクを使用することで処理能力の向上が得られるものと思われる。
　また、メモリ使用量についてもクライアント数等に関係なく一定量使用していることが分かる。

ｂ．実検索／トランザクションでのベンチマークテスト

PostgreSQL にて郵便番号データを用いて実トランザクションのベンチマークテストを実施。
対象データは約１２万件で処理した。
評価項目として、以下の SQL 文は発行し各処理時間のデータ収集を１度ずつ行った。

＜ベンチマーク結果＞

SQL	PostgreSQL		
	Connect	Query	Total
文字列完全一致	930	390	1,320
文字列前方一致	940	16,910	17,850
文字列部分一致	930	4,730	5,660
AND 条件検索	940	4,720	5,660
全件挿入	930	15,210	16,140
全件更新	1,480	2,250	3,730
全件削除	940	880	1,820
ﾃｰﾌﾞﾙ結合	940	123,580	124,520

(単位：msec)

評価を行う上で使用した SQL 文およびテーブル構成の詳細は以下のとうりである。

＜使用 SQL 文＞

	SQL 文
文字列完全一致	Select * from yubin_table1 where yubin_no = '1300014' ;
文字列前方一致	Select * from yubin_table1 where kanji_cho like '新町%' ;
文字列部分一致	Select * from yubin_table1 where kanji_cho like '%新町%' ;
AND 条件検索	Select * from yubin_table1 where kanji_ken like '%京%' and kanji_si like '%大%' and kanji_cho like '%本町%' ;
全件挿入	insert into yubin_table1_cp (select * from yubin_table1 where kanji_ken like '大分県%' ;
全件更新	update yubin_table1_cp set kanji_ken = 'oisa' ;
全件削除	delete from yubin_table1_cp ;
ﾃｰﾌﾞﾙ結合	select a.yubin_no , b.kanji_ken from yubin_table1 b , yubin_table3 a where a.yubin2_flg = '0' and a.yubin_no = b.yubin_no ;

＜yubin_table1＞

No	フィールド名	属性	キー	索引	備考
1	Yubin_no	Varchar(7)		○	郵便番号
2	Kanji_ken	Varchar(8)		○	都道府県名（漢字）
3	Kanji_si	Varchar(24)		○	市町村区名（漢字）
4	Kanji_cho	Varchar(74)		○	町域以下（漢字）

＜yubin_table2＞

No	フィールド名	属性	キー	索引	備考
1	yubin_no	Varchar(7)		○	郵便番号
2	kana_ken	Varchar(16)		○	都道府県名（カナ）
3	kana_si	Varchar(48)		○	市町村区名（カナ）
4	kana_cho	Varchar(148)		○	町域以下（カナ）

＜yubin_table3＞

No	フィールド名	属性	キー	索引	備考
1	yubin_no	Varchar(7)		○	郵便番号
2	kanji_ken	Varchar(8)		○	都道府県名（漢字）　初期値 “oisa”
3	kanji_si	Varchar(24)		○	市町村区名（漢字）　初期値 “oisa”
4	kanji_cho	Varchar(74)		○	町域以下（漢字）　初期値 “oisa”
5	kana_ken	Varchar(16)		○	都道府県名（カナ）　初期値 “oisa”
6	kana_si	Varchar(48)		○	市町村区名（カナ）　初期値 “oisa”
7	kana_cho	Varchar(148)		○	町域以下（カナ）　初期値 “oisa”
8	cho2_flg	Varchar(1)			町域重複フラグ
9	chome_flg	Varchar(1)			丁目フラグ
10	yubin2_flg	Varchar(1)			郵便番号重複フラグ

※町域重複フラグ：一町域が二以上の新郵便番号で表される場合の表示
(「1」は該当、「0」は該当せず)
※丁目フラグ：丁目を有する町域の場合の表示
(「1」は該当、「0」は該当せず)
※郵便番号重複フラグ：一つの新郵便番号で二以上の町域を表す場合の表示
(「1」は該当、「0」は該当せず)

i). 評価分析

ベンチマークの結果より、PostgreSQL ではＤＢへのコネクション時間は“全件更新”のパターンを除いてほぼ同一時間を要している。しかしＱｕｅｒｙ処理においては“文字列前方一致”“全件挿入”“テーブル結合”は他の処理に比べ時間を要しており特に“テーブル結合”においては顕著である。
このことから、PostgreSQL については単純操作についてはかなり速い結果が得られるものの、複雑操作は反対にかなりの時間を要するものと思われる。

c．総合評価

以上のように、仮想クライアントおよび実検索／トランザクションでのベンチマークテスト結果を踏まえて、ある程度は今回の問題である小規模データベースの実業務運用の解となり得るものではないかと思われる。
但し、今回テスト評価はそれぞれ１度づつしか行っておらず、必ずしも今回結果が正しい評価とは言い切れないため、さらなる評価は必要である。

５．おわりに

本研究部会の勉強会は、Linux と遊休資産のハードウェアの組み合わせで何が出来るのか？、どれくらい出来るのか？に着目して活動を行ってきた。その結果、“小規模なデータベースで有れば実業務運用も可能ではないか”という事を、メンバ全員が感じている所である。

これらの選定や実現手法については、これまでに縷々述べてきたが、下記のような注意点が挙げられる。

(1).Linux の実用性は高く、業務運用にも利用可能である事が感じられた。
しかしながら、Linux の導入から調整までを行う上で必要な、マニュアルや資料が十分とは言い難い事と、マニュアル通りにはならない事等、ユーザ自身での解決力や情報収集力が必要と感じた。
最近では、ハードウェアベンダーからも **Linux** 導入済のサーバシステムが発売されており、これらを利用する事で、手法とノウハウの蓄積も可能となるのではないだろうか。

(2).システムは、導入して稼動すれば完了ではない。
システムは運用していく上で、その利用形態により進化と拡張が必要となってくると思われる。本部会では、導入と評価にとどまっているが、業務運用を前提とした場合にこれらは重要なキーワードとなる為、各種ソフトウェアの選定には注意が必要と思われる。

(3).遊休資産と言われる、第一線を退いたハードウェアを利用する(出来る)事は、本部会でも確認出来た。しかし、構築するシステムによっては多くのハードウェア資源を必要とする事も活動中に実感した。
構築するシステムにより、メインメモリ増設やハードディスク容量等、資源の見直しが必要である。

(4).今回の検証では、性能についての一般的な評価は、時間的・物理的な制約により保留とした。適用にあたっては個別の業務用件にてシステムのチューニングも含めて検討する必要がある。

本研究部会では、システム構築の一部及び、評価・検証の一端を行っただけでは有るが、メンバ一人ひとりが、持てる知識を出し合い、お互いに協力しての成果である。また、無料のメールグループサービス(http://www.egroups.co.jp)を活用して情報の一元化を図る事により、効率的な研究部会の運営が出来た。これらの経験を、今後の活躍に活かして行きたい。

最後に、本研究部会の開催に際し、ご理解とご協力を頂きました関係各社の皆様方にお礼を申し上げまして、本研究部会の活動を終了致します。　有難う御座いました。

6．コメント

【高橋　昭光　三井造船システム技研 株式会社】

Unix の知識が多少あった為、Linux への抵抗感は有りませんでした。私自身、数年前にテスト的に遊休パソコンを利用し Linux を導入した経験が有りましたが、信頼性・実用性は度外視でした。
今回の活動で、「Linux は使える」という事を実感すると共に、RDBMS 等のアプリケーションも十分活用できると感じました。
また、業務多忙のなか研究部会に開催に際しまして、ご理解とご協力を頂きましたメンバの方々にお礼申し上げます。　有難う御座いました。

【西河原　洋一　株式会社 オーイーシー】

過去、自分で Linux のセットアップをして自分なりに理解していたが、あやふやだった部分や触れていない部分の見直しが出来たのでいい機会でした。
また、活動中盤より業務出張で参加出来なくなってしまい非常に残念でしたが、今回の経験を基に、今後機会があれば Linux を使って行きたいと思います。

【亀井　浩　株式会社 富士通大分ソフトウェアラボラトリ】

DBサーバとしてのLinuxをテーマとするにあたり、本当に業務に適用できるレベルにあるのか、実際にさわって見極めてみたいという興味が、個人的にはありました。
時間的な制約もあり、結論には至っていませんが、少なくとも運用時の安定性については、巷での評判どおりに良好なものであると感じました。
また、適切にチューニングすることによって色々な場面で適用できそうな、奥の深さも感じました。
もちろん、大規模業務に適用する場合には、障害発生時のリカバリをどうするか、クラスタリングは可能なのか、スケーラビリティはどれほどのものなのか、などの問題はあります。
しかし、オープンソースソフトウェアの世界は、レベルアップのサイクルも早い為、それらの課題も早晩に解決されるものと期待しています。

【安井　正樹　株式会社 富士通大分ソフトウェアラボラトリ】

インストーラおよび GNOME で代表される統合デスクトップ環境の整備に、オープンソースの将来性を感じた。今後も、オープンソース・コミュニティに注視したいと考えている。

【加藤　匡　鶴崎海陸運輸 株式会社】

当研究会を通じて得た、Ｌｉｎｕｘの知識および社外の方との交流を今後の業務に活かしていきたいと思います。

【宗安　隆行　三井造船システム技研 株式会社】

今回、初めてＬｉｎｕｘを操作しましたが安定性や操作性には、あまり不快感を感じませんでした。しかし、対応ソフトや書籍（情報）が少ないこと等から一般ユーザーに普及するのは、まだこれからだと思いました。

【糸野　憲和　株式会社 長嶋不動産鑑定事務所】

部会長さんはじめ、部会員の皆さん、本当にお疲れさまでした。
研究部会を通じて、課題へ熱心に取り組む姿勢は、技術研究会としての目的を十分に果たすことができたと思います。
本研究部会で一緒に興味深く取り組むことができ、良い経験をさせていただきました。

7．部会開催履歴

第1回　平成１２年　7月１２日　１５：００～
　　平成１２年度「技術研究会」実施要領全体説明
　　Ｌｉｎｕｘ部会メンバ顔合わせ

第2回　平成１２年　7月２７日　１４：００～
　　テーマ選定及び今後のスケジュール決定

第3回　平成１２年　8月１１日　１３：００～
　　Ｌｉｎｕｘ　ＤＢサーバのセットアップ

第4回　平成１２年　8月３０日　１３：００～
　　Ｌｉｎｕｘ　ＤＢサーバの再構築

第5回　平成１２年　9月１３日　１３：００～
　　Ｌｉｎｕｘ版　Ｏｒａｃｌｅ導入

第6回　平成１２年１０月１１日　１３：００～
　　ＰｏｓｔｇｒｅＳＱＬ・Ｏｒａｃｌｅ評価

第7回　平成１２年１１月　8日　１３：００～
　　ベンチマークテストの準備

第8回　平成１２年１１月２９日　１３：３０～
　　性能測定(ベンチマークテスト)

第9回　平成１３年　1月２５日　１３：００～
　　作成論文の編集及び、プレゼンテーション資料作成

引用文献

多比羅悟（1999）　『図解でわかる Linux サーバーのすべて』 日本実業出版社　P.188-223
篠田典良（2000）『リファレンス Linux による Oracle8i システム管理ガイド』リックテレコム
R.Card,E.Dumas,F.Mevel（1999）『Linux2.0 カーネルブック』オーム社
糸魚川茂夫（1999）『FreeBSD/Linux で使う PostgreSQL 詳細』オーム社
藤田泰徳・山崎文則（2000）『Apache サーバと PostgreSQL でデータベース オープンソースを使ったデータベースの構築』セレンディップ
小野哲・菊地貴裕・伊藤宏道（1999）『Oracle8 for Linux データベース導入実践ガイド』技術評論社
R.J.Yarger,G.Reese,T.King（2000）『MySQL＆mSQL』オライリー・ジャパン

笠原規男（2000）”Linux データベース・レシピ 第６回 定番、アドレス帳を作る”
　Linux Japan、五橋研究所、P.154-162、JANUARY,2000
笠原規男（2000）”Linux データベース・レシピ 第８回 メーラーだって RDB”
　Linux Japan、五橋研究所、P.149-157、March,2000
笠原規男（2000）”Linux データベース・レシピ WINGZ を試食する”
　Linux Japan、五橋研究所、P.125-131、November,2000
笠原規男（1999）” Linux データベース・レシピ 第４回 宝石探しの旅（１）”
　Linux Japan、五橋研究所、P.139-146、November,1999
笠原規男（1999）” Linux データベース・レシピ 第３回 ブックマークデータベースを作る（３）”
　Linux Japan、五橋研究所、P.137-144、October,1999
ウチエ・オグブジ（1999）”Linux による Oracle 活用 その基礎を知る Linux で企業データベースを用いるための基礎ガイド” Linux WORLD、IDG コミュニケーションズ、P.162-166、保存版第３弾
小野哲・鈴木大岳（2000）”Oracle8i による DB サーバ構築 第２回 Oracle8i をインストールして動作を確認する” 日経 Linux、日経 BP 社、P.151-160、JULY,2000
片岡裕生（1999）”探索フリーソフト PostgreSQL を GUI で操作 PgAccess” 日経 Linux、日経 BP 社、P.124-127、November,1999
吉川和巳（1999）”徹底追跡！Linux 対応データベース最新情報” Linux WORLD、IDG コミュニケーションズ、P.74-83、保存版第１弾
ロブ・メルダ（1999）”世界中の Linux ユーザーから支持を集めるニュース・サイトの構築プロセス” Linux WORLD、IDG コミュニケーションズ、P.144-148、保存版第１弾
石井達夫・村上毅（1999）”オープンソース DBMS の実力を探る” 日経 Linux、日経 BP 社、P.104-127、October,1999

日経 Linux、日経 BP 社、付録 CD-ROM、JULY,2000
　Oracle8i Workgroup Server for Linux R8.1.5 （90 日間トライアル版）
　Official Red Hat Linux 6.2J FTP 版

http://www.worldserver.com/mm.mysql/performance/
←JDBCBench を用いたシミュレーション結果、JDBCBench.java のダウンロード

http://www.unify.com/jp/products/dataserver/linux.html
←JDBCBench を用いたシミュレーション結果

http://nos.nikkeibp.co.jp/9911toku2/
←SRA 社が行った PostgreSQL のベンチマークテスト

http://www.postal.mpt.go.jp/newnumber/down_2.htm
←郵便番号データのダウンロード

ftp://ftp.lab.kdd.co.jp/Linux/java-linux/
←jre_1.1.6-v5-glibc-x86.tar.gz（BlackDown のミラーサイト）

http://www.pugly.juice.or.jp/pugly/postgres/jdbc.html
←postgresql.jar（PostgreSQL の JDBC ドライバ）

http://technet.oracle.co.jp/software/db_connect/jdbc/81601/pr_jdbc81601.html
←classes12.zip（Oracle の JDBC ドライバ）

http://www.interwiz.koganei.tokyo.jp/software/PsqlODBC/
←PostgreSQL の ODBC ドライバ（日本語版）

ftp://ftp.postgresql.org/pub/odbc/latest/postdrv.exe
←PostgreSQL の ODBC ドライバ（英語版）

http://technet.oracle.co.jp/software/db_connect/odbc/8162/pr_odbc8162.html
←Oracle の ODBC ドライバ

http://java.sun.com/products/jdk/1.2/ja/jre/download-windows.html
←SUN の JRE（国際化版）

別添１．１．**仮想クライアントでのベンチマークテスト用 Java プログラム（PostgreSQL 版）**

```
/*
 *  This is a sample implementation of the Transaction Processing Performance
 *  Council Benchmark B coded in Java and ANSI SQL2.
 */

import java.sql.*;

public class JDBCBench
{

    /* tpc bm b scaling rules */
    public static int tps       =      1;  /* the tps scaling factor: here it is 1 */
    public static int nbranches =      1;  /* number of branches in 1 tps db       */
    public static int ntellers  =     10;  /* number of tellers in  1 tps db       */
    public static int naccounts =  100000;  /* number of accounts in 1 tps db       */
    public static int nhistory  =  864000;  /* number of history recs in 1 tps db   */

    public final static int TELLER  = 0;
    public final static int BRANCH  = 1;
    public final static int ACCOUNT = 2;



    private Connection Conn = null;

    int failed_transactions = 0;
    int transaction_count = 0;
    static int n_clients = 10;
    static int n_txn_per_client = 10;
    long start_time = 0;

    static boolean verbose = false;

    MemoryWatcherThread MemoryWatcher;

    /* main program,    creates a 1-tps database:  i.e. 1 branch, 10 tellers,...
     *                    runs one TPC BM B transaction
     */

    public static void main(String[] Args)
    {
      String DriverName = "";
      String DBUrl = "";
//  mente-000 yasui(OSL) 2000/11/12
      String  user = "" , pass="";
//  mente-999 yasui(OSL) 2000/11/12

      boolean initialize_dataset = false;

      for (int i = 0; i < Args.length; i++) {
        if (Args[i].equals("-clients")) {
```

```
            if (i + 1 < Args.length) {
                i++;
                n_clients = Integer.parseInt(Args[i]);
            }

        }
        else if (Args[i].equals("-driver")) {
            if (i + 1 < Args.length) {
                i++;
                DriverName = Args[i];
            }
        }
        else if (Args[i].equals("-url")) {
            if (i + 1 < Args.length) {
                i++;
                DBUrl = Args[i];
            }
        }

// mente-000 yasui(OSL) 2000/11/12
        else if (Args[i].equals("-user")) {
            if (i + 1 < Args.length) {
                i++;
                user = Args[i];
            }
        }
        else if (Args[i].equals("-pass")) {
            if (i + 1 < Args.length) {
                i++;
                pass = Args[i];
            }
        }
// mente-999 yasui(OSL) 2000/11/12

        else if (Args[i].equals("-tpc")) {
            if (i + 1 < Args.length) {
                i++;
                n_txn_per_client = Integer.parseInt(Args[i]);
            }
        }
        else if (Args[i].equals("-init")) {
            initialize_dataset = true;
        }
        else if (Args[i].equals("-v")) {
            verbose = true;
        }
      }

      if (DriverName.length() == 0 || DBUrl.length() == 0) {
        System.out.println("usage: java JDBCBench -driver [driver_class_name] -url [url_to_db]
[-v] [-init] [-tpc n] [-clients]");
        System.out.println();
```

```
        System.out.println("-v          verbose error messages");
        System.out.println("-init       initialize the tables");
        System.out.println("-tpctransactions per client");
        System.out.println("-clients    number of simultaneous clients");
        System.exit(-1);
      }

      System.out.println("*************************************************************");
      System.out.println("* JDBCBench v1.0                                            *");
      System.out.println("*************************************************************");
      System.out.println();
      System.out.println("Driver: " + DriverName);
      System.out.println("URL:" + DBUrl);
      System.out.println();
      System.out.println("Number of clients: " + n_clients);
      System.out.println("Number of transactions per client: " + n_txn_per_client);
      System.out.println();

      try {
        Class.forName(DriverName);
// mente-000 yasui(OSL) 2000/11/12
 //       Connection C = DriverManager.getConnection(DBUrl);
         Connection C = DriverManager.getConnection(DBUrl,user,pass);
// mente-999 yasui(OSL) 2000/11/12

        JDBCBench Me = new JDBCBench(C, initialize_dataset);
      }
      catch (Exception E) {
        System.out.println(E.getMessage());
        E.printStackTrace();
      }
    }

    public JDBCBench(Connection C, boolean init)
    {
        try {
            Conn = C;
            if (init) {
                System.out.print("Initializing dataset...");
                createDatabase();
                System.out.println("done.\n");
            }
            System.out.println("* Starting Benchmark Run *");
            MemoryWatcher = new MemoryWatcherThread();
            MemoryWatcher.start();

            start_time = System.currentTimeMillis();

            for (int i = 0; i < n_clients; i++) {
                Thread Client = new ClientThread(n_txn_per_client);
                Client.start();
            }
```

```
        }
        catch (Exception E) {
            System.out.println(E.getMessage());
            E.printStackTrace();
        }
    }

    public void reportDone()
    {
        n_clients--;

        if (n_clients <= 0) {
            MemoryWatcher.stop();

            long end_time = System.currentTimeMillis();
            double completion_time = ((double)end_time - (double)start_time) / 1000;
            System.out.println("* Benchmark finished *");
            System.out.println("\n* Benchmark Report *" );
            System.out.println("--------------------\n");
            System.out.println("Time to execute " +  transaction_count + " transactions: "  +
completion_time + " seconds.");
            System.out.println("Max/Min memory usage: " + MemoryWatcher.max + " / " +
MemoryWatcher.min + " kb");
            System.out.println(failed_transactions + " / " + transaction_count + " failed to
complete.");
            System.out.println("Transaction    rate:    "    +    (transaction_count    -
failed_transactions) / completion_time + " txn/sec.");
        }

    }

    public synchronized void incrementTransactionCount()
    {
        transaction_count++;
    }

    public synchronized void incrementFailedTransactionCount()
    {
        failed_transactions++;
    }


    /*
     *  createDatabase() - Creates and Initializes a scaled database.
     */

    void createDatabase() throws Exception
    {

        try {
            Statement Stmt = Conn.createStatement();
```

```
            String Query = "CREATE TABLE branches (";
               Query+= "Bid         INT NOT NULL, PRIMARY KEY(Bid), ";
               Query+= "Bbalance    INT,";
               Query+= "filler      CHAR(88))"; /* pad to 100 bytes */
            Stmt.executeUpdate(Query);
            Stmt.clearWarnings();

            Query = "CREATE TABLE tellers ( ";
            Query+= "Tid         INT NOT NULL, PRIMARY KEY(Tid),";
            Query+= "Bid         INT,";
            Query+= "Tbalance    INT,";
            Query+= "filler      CHAR(84))"; /* pad to 100 bytes */

            Stmt.executeUpdate(Query);
            Stmt.clearWarnings();

            Query = "CREATE TABLE accounts ( ";
            Query+= "Aid         INT NOT NULL, PRIMARY KEY(Aid), ";
            Query+= "Bid         INT, ";
            Query+= "Abalance    INT, ";
            Query+= "filler      CHAR(84))";      /* pad to 100 bytes */

            Stmt.executeUpdate(Query);
            Stmt.clearWarnings();

            Query = "CREATE TABLE history ( ";
            Query+= "Tid         INT, ";
            Query+= "Bid         INT, ";
            Query+= "Aid         INT, ";
            Query+= "delta       INT, ";
            Query+= "time        TIMESTAMP, ";
            Query+= "filler      CHAR(22))"; /* pad to 50 bytes  */

            Stmt.executeUpdate(Query);
            Stmt.clearWarnings();

            /* prime database using TPC BM B scaling rules.
             *  Note that for each branch and teller:
             *      branch_id = teller_id  / ntellers
             *      branch_id = account_id / naccounts
             */

            for (int i = 0; i < nbranches * tps; i++) {
                Query = "INSERT INTO branches(Bid,Bbalance) VALUES (" + i + ",0)";
                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();
            }
            for (int i = 0; i < ntellers * tps; i++) {
                Query = "INSERT INTO tellers(Tid,Bid,Tbalance) VALUES (" + i + "," + i / ntellers
+ ",0)";
                Stmt.executeUpdate(Query);
```

```
                Stmt.clearWarnings();
            }
            for (int i = 0; i < naccounts*tps; i++) {
                Query = "INSERT INTO accounts(Aid,Bid,Abalance) VALUES (" + i + "," + i /
naccounts + ",0)";
                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();
            }
// mente-000 yasui(OSL) 2000/11/12
            Stmt.close();
// mente-999 yasui(OSL) 2000/11/12

        }
        catch (Exception E) {
            System.out.println(E.getMessage());
            E.printStackTrace();
        }

    }           /* end of CreateDatabase    */

    public static int getRandomInt(int lo, int hi)
    {
          int ret = 0;

          ret = (int)(Math.random() * (hi - lo + 1));
          ret += lo;

          return ret;
    }

    public static int getRandomID(int type)
    {
        int min, max, num;

        max = min = 0;
        num = naccounts;

        switch(type) {
        case TELLER:
                min += nbranches;
                num = ntellers;
                /* FALLTHROUGH */
        case BRANCH:
                if (type == BRANCH)
                        num = nbranches;
                min += naccounts;
                /* FALLTHROUGH */
        case ACCOUNT:
                max = min + num - 1;
        }
        return (getRandomInt(min, max));
    }
```

```
    class ClientThread extends Thread
    {
        int ntrans = 0;

        public ClientThread(int number_of_txns)
        {
            ntrans = number_of_txns;
        }

        public void run()
        {
            while (ntrans-- > 0) {
  /*
                int account = JDBCBench.getRandomID(ACCOUNT);
                int branch  = JDBCBench.getRandomID(BRANCH);
                int teller  = JDBCBench.getRandomID(TELLER);
*/
                int account = JDBCBench.getRandomInt(0,naccounts-1);
                int branch  = JDBCBench.getRandomInt(0,nbranches-1);
                int teller  = JDBCBench.getRandomInt(0,ntellers-1);

                int delta   = JDBCBench.getRandomInt(0,1000);

                doOne(branch, teller, account, delta);
                incrementTransactionCount();
            }
            reportDone();
        }

        /*
         *  doOne() - Executes a single TPC BM B transaction.
         */

        int doOne(int bid, int tid, int aid, int delta)
        {
            try {
                Statement Stmt = Conn.createStatement();

                String Query = "UPDATE accounts ";
                       Query+= "SET     Abalance = Abalance + " + delta + " ";
                       Query+= "WHERE   Aid = " + aid;

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

                Query = "SELECT Abalance ";
                Query+= "FROM   accounts ";
                Query+= "WHERE  Aid = " + aid;

                ResultSet RS = Stmt.executeQuery(Query);
                Stmt.clearWarnings();
```

```
                int aBalance = 0;

                while (RS.next()) {
                    aBalance = RS.getInt(1);
                }

                Query = "UPDATE tellers ";
                Query+= "SET    Tbalance = Tbalance + " + delta + " ";
                Query+= "WHERE  Tid = " + tid;

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

                Query = "UPDATE branches ";
                Query+= "SET    Bbalance = Bbalance + " + delta + " ";
                Query+= "WHERE  Bid = " + bid;

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

                Query = "INSERT INTO history(Tid, Bid, Aid, delta) ";
                Query+= "VALUES (";
                Query+= tid + ",";
                Query+= bid + ",";
                Query+= aid + ",";
                Query+= delta + ")";

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

// mente-000 yasui(OSL) 2000/11/12
            Stmt.close();
                    RS.close();
// mente-999 yasui(OSL) 2000/11/12

                return aBalance;
            }
            catch (SQLException E) {
                if (verbose) {
                    System.out.println("Transaction failed: " + E.getMessage());
                    E.printStackTrace();
                }
                incrementFailedTransactionCount();
            }
            return 0;

        }           /* end of DoOne         */
```

```
    }

    class MemoryWatcherThread extends Thread
    {
        long min = 0;
        long max = 0;

        public void run()
        {
            min = Runtime.getRuntime().freeMemory();

            for (;;) {
                long currentFree = Runtime.getRuntime().freeMemory();
                long currentAlloc = Runtime.getRuntime().totalMemory();
                long used = currentAlloc - currentFree;

                if (used < min)
                    min = used;
                if (used > max)
                    max = used;

                try {
                    sleep(100);
                }
                catch (InterruptedException E) {}
            }
         }
    }



}
```

別添１．２．**仮想クライアントでのベンチマークテスト用 Java プログラム（Oracle8i 版）**

```
/*
 *  This is a sample implementation of the Transaction Processing Performance
 *  Council Benchmark B coded in Java and ANSI SQL2.
 */

import java.sql.*;

public class JDBCBench
{

    /* tpc bm b scaling rules */
    public static int tps       =       1;  /* the tps scaling factor: here it is 1 */
    public static int nbranches =       1;  /* number of branches in 1 tps db       */
    public static int ntellers  =      10;  /* number of tellers in  1 tps db       */
    public static int naccounts =  100000;  /* number of accounts in 1 tps db       */
    public static int nhistory  =  864000;  /* number of history recs in 1 tps db   */

    public final static int TELLER  = 0;
    public final static int BRANCH  = 1;
    public final static int ACCOUNT = 2;



    private Connection Conn = null;

    int failed_transactions = 0;
    int transaction_count = 0;
    static int n_clients = 10;
    static int n_txn_per_client = 10;
    long start_time = 0;

    static boolean verbose = false;

    MemoryWatcherThread MemoryWatcher;

    /* main program,    creates a 1-tps database:  i.e. 1 branch, 10 tellers,...
     *                    runs one TPC BM B transaction
     */

    public static void main(String[] Args)
    {
      String DriverName = "";
      String DBUrl = "";
//  mente-000 yasui(OSL) 2000/11/12
      String  user = "" , pass="";
//  mente-999 yasui(OSL) 2000/11/12
      boolean initialize_dataset = false;

      for (int i = 0; i < Args.length; i++) {
        if (Args[i].equals("-clients")) {
```

```
            if (i + 1 < Args.length) {
                i++;
                n_clients = Integer.parseInt(Args[i]);
            }
        }
        else if (Args[i].equals("-driver")) {
            if (i + 1 < Args.length) {
                i++;
                DriverName = Args[i];
            }
        }
        else if (Args[i].equals("-url")) {
            if (i + 1 < Args.length) {
                i++;
                DBUrl = Args[i];
            }
        }
// mente-000 yasui(OSL) 2000/11/12
        else if (Args[i].equals("-user")) {
            if (i + 1 < Args.length) {
                i++;
                user = Args[i];
            }
        }
        else if (Args[i].equals("-pass")) {
            if (i + 1 < Args.length) {
                i++;
                pass = Args[i];
            }
        }
// mente-999 yasui(OSL) 2000/11/12
        else if (Args[i].equals("-tpc")) {
            if (i + 1 < Args.length) {
                i++;
                n_txn_per_client = Integer.parseInt(Args[i]);
            }
        }
        else if (Args[i].equals("-init")) {
            initialize_dataset = true;
        }
        else if (Args[i].equals("-v")) {
            verbose = true;
        }
      }

      if (DriverName.length() == 0 || DBUrl.length() == 0) {
        System.out.println("usage: java JDBCBench -driver [driver_class_name] -url [url_to_db]
[-v] [-init] [-tpc n] [-clients]");
        System.out.println();
        System.out.println("-v          verbose error messages");
        System.out.println("-init       initialize the tables");
        System.out.println("-tpctransactions per client");
```

```
        System.out.println("-clients    number of simultaneous clients");
        System.exit(-1);
      }

      System.out.println("*********************************************************");
      System.out.println("* JDBCBench v1.0                                        *");
      System.out.println("*********************************************************");
      System.out.println();
      System.out.println("Driver: " + DriverName);
      System.out.println("URL:" + DBUrl);
      System.out.println();
      System.out.println("Number of clients: " + n_clients);
      System.out.println("Number of transactions per client: " + n_txn_per_client);
      System.out.println();

      try {
        Class.forName(DriverName);

// mente-000 yasui(OSL) 2000/11/12
//      Connection C = DriverManager.getConnection(DBUrl);
        Connection C = DriverManager.getConnection(DBUrl,user,pass);
// mente-999 yasui(OSL) 2000/11/12

        JDBCBench Me = new JDBCBench(C, initialize_dataset);
      }
      catch (Exception E) {
        System.out.println(E.getMessage());
        E.printStackTrace();
      }
    }

    public JDBCBench(Connection C, boolean init)
    {
        try {
            Conn = C;
            if (init) {
                System.out.print("Initializing dataset...");
                createDatabase();
                System.out.println("done.\n");
            }
            System.out.println("* Starting Benchmark Run *");
            MemoryWatcher = new MemoryWatcherThread();
            MemoryWatcher.start();

            start_time = System.currentTimeMillis();

            for (int i = 0; i < n_clients; i++) {
                Thread Client = new ClientThread(n_txn_per_client);
                Client.start();
            }
        }
        catch (Exception E) {
```

```
            System.out.println(E.getMessage());
            E.printStackTrace();
        }
    }

    public void reportDone()
    {
        n_clients--;

        if (n_clients <= 0) {
            MemoryWatcher.stop();

            long end_time = System.currentTimeMillis();
            double completion_time = ((double)end_time - (double)start_time) / 1000;
            System.out.println("* Benchmark finished *");
            System.out.println("\n* Benchmark Report *" );
            System.out.println("--------------------\n");
            System.out.println("Time to execute " +  transaction_count + " transactions: "  +
completion_time + " seconds.");
            System.out.println("Max/Min memory usage: " + MemoryWatcher.max + " / " +
MemoryWatcher.min + " kb");
            System.out.println(failed_transactions + " / " + transaction_count + " failed to
complete.");
            System.out.println("Transaction    rate:    "    +    (transaction_count    -
failed_transactions) / completion_time + " txn/sec.");
        }

    }

    public synchronized void incrementTransactionCount()
    {
        transaction_count++;
    }

    public synchronized void incrementFailedTransactionCount()
    {
        failed_transactions++;
    }


    /*
     *  createDatabase() - Creates and Initializes a scaled database.
     */

    void createDatabase() throws Exception
    {

        try {
            Statement Stmt = Conn.createStatement();

            String Query = "CREATE TABLE branches (";
               Query+= "Bid         INT NOT NULL, PRIMARY KEY(Bid), ";
```

```
            Query+= "Bbalance    INT,";
            Query+= "filler      CHAR(88))"; /* pad to 100 bytes */


        Stmt.executeUpdate(Query);
        Stmt.clearWarnings();

        Query = "CREATE TABLE tellers ( ";
        Query+= "Tid         INT NOT NULL, PRIMARY KEY(Tid),";
        Query+= "Bid         INT,";
        Query+= "Tbalance    INT,";
        Query+= "filler      CHAR(84))"; /* pad to 100 bytes */

        Stmt.executeUpdate(Query);
        Stmt.clearWarnings();

        Query = "CREATE TABLE accounts ( ";
        Query+= "Aid         INT NOT NULL, PRIMARY KEY(Aid), ";
        Query+= "Bid         INT, ";
        Query+= "Abalance    INT, ";
        Query+= "filler      CHAR(84))";      /* pad to 100 bytes */

        Stmt.executeUpdate(Query);
        Stmt.clearWarnings();

        Query = "CREATE TABLE history ( ";
        Query+= "Tid         INT, ";
        Query+= "Bid         INT, ";
        Query+= "Aid         INT, ";
        Query+= "delta       INT, ";
        Query+= "time        date, ";
        Query+= "filler      CHAR(22))"; /* pad to 50 bytes  */

        Stmt.executeUpdate(Query);
        Stmt.clearWarnings();

        /* prime database using TPC BM B scaling rules.
         *  Note that for each branch and teller:
         *      branch_id = teller_id  / ntellers
         *      branch_id = account_id / naccounts
         */

        for (int i = 0; i < nbranches * tps; i++) {
            Query = "INSERT INTO branches(Bid,Bbalance) VALUES (" + i + ",0)";
            Stmt.executeUpdate(Query);
            Stmt.clearWarnings();
        }
        for (int i = 0; i < ntellers * tps; i++) {
            Query = "INSERT INTO tellers(Tid,Bid,Tbalance) VALUES (" + i + "," + i / ntellers
+ ",0)";
            Stmt.executeUpdate(Query);
            Stmt.clearWarnings();
```

```
            }
            for (int i = 0; i < naccounts*tps; i++) {
                Query = "INSERT INTO accounts(Aid,Bid,Abalance) VALUES (" + i + "," + i /
naccounts + ",0)";
                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();
            }
// mente-000 yasui(OSL) 2000/11/12
            Stmt.close();
// mente-999 yasui(OSL) 2000/11/12

        }
        catch (Exception E) {
            System.out.println(E.getMessage());
            E.printStackTrace();
        }

    }           /* end of CreateDatabase    */

    public static int getRandomInt(int lo, int hi)
    {
          int ret = 0;

          ret = (int)(Math.random() * (hi - lo + 1));
          ret += lo;

          return ret;
    }

    public static int getRandomID(int type)
    {
        int min, max, num;

        max = min = 0;
        num = naccounts;

        switch(type) {
        case TELLER:
                min += nbranches;
                num = ntellers;
                /* FALLTHROUGH */
        case BRANCH:
                if (type == BRANCH)
                        num = nbranches;
                min += naccounts;
                /* FALLTHROUGH */
        case ACCOUNT:
                max = min + num - 1;
        }
        return (getRandomInt(min, max));
    }
```

```
    class ClientThread extends Thread
    {
        int ntrans = 0;

        public ClientThread(int number_of_txns)
        {
            ntrans = number_of_txns;
        }

        public void run()
        {
            while (ntrans-- > 0) {
 /*
                int account = JDBCBench.getRandomID(ACCOUNT);
                int branch  = JDBCBench.getRandomID(BRANCH);
                int teller  = JDBCBench.getRandomID(TELLER);
*/
                int account = JDBCBench.getRandomInt(0,naccounts-1);
                int branch  = JDBCBench.getRandomInt(0,nbranches-1);
                int teller  = JDBCBench.getRandomInt(0,ntellers-1);



                int delta   = JDBCBench.getRandomInt(0,1000);

                doOne(branch, teller, account, delta);
                incrementTransactionCount();
            }
            reportDone();
        }

        /*
         *  doOne() - Executes a single TPC BM B transaction.
         */

        int doOne(int bid, int tid, int aid, int delta)
        {
            try {
                Statement Stmt = Conn.createStatement();

                String Query = "UPDATE accounts ";
                       Query+= "SET     Abalance = Abalance + " + delta + " ";
                       Query+= "WHERE   Aid = " + aid;

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

                Query = "SELECT Abalance ";
                Query+= "FROM   accounts ";
                Query+= "WHERE  Aid = " + aid;

                ResultSet RS = Stmt.executeQuery(Query);
```

```
                Stmt.clearWarnings();

                int aBalance = 0;

                while (RS.next()) {
                    aBalance = RS.getInt(1);
                }

                Query = "UPDATE tellers ";
                Query+= "SET    Tbalance = Tbalance + " + delta + " ";
                Query+= "WHERE  Tid = " + tid;

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

                Query = "UPDATE branches ";
                Query+= "SET    Bbalance = Bbalance + " + delta + " ";
                Query+= "WHERE  Bid = " + bid;

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

                Query = "INSERT INTO history(Tid, Bid, Aid, delta) ";
                Query+= "VALUES (";
                Query+= tid + ",";
                Query+= bid + ",";
                Query+= aid + ",";
                Query+= delta + ")";

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

// mente-000 yasui(OSL) 2000/11/12
            Stmt.close();
                    RS.close();
// mente-999 yasui(OSL) 2000/11/12


                return aBalance;
            }
            catch (SQLException E) {
                if (verbose) {
                    System.out.println("Transaction failed: " + E.getMessage());
                    E.printStackTrace();
                }
                incrementFailedTransactionCount();
            }
            return 0;

        }           /* end of DoOne         */
```

```
    }

    class MemoryWatcherThread extends Thread
    {
        long min = 0;
        long max = 0;

        public void run()
        {
            min = Runtime.getRuntime().freeMemory();

            for (;;) {
                long currentFree = Runtime.getRuntime().freeMemory();
                long currentAlloc = Runtime.getRuntime().totalMemory();
                long used = currentAlloc - currentFree;

                if (used < min)
                    min = used;
                if (used > max)
                    max = used;

                try {
                    sleep(100);
                }
                catch (InterruptedException E) {}
            }
         }
    }



}
```

別添１．３．**実クライアントでのベンチマークテスト用** Java **プログラム（**PostgreSQL **版・参照系）**

```
import java.io.*;
import java.util.*;

import java.sql.*;
import java.util.Date;

public class test {

 public test() {

StringTokenizer st_token =null;
File file = new File("./sg.txt");
try {

  FileInputStream fis = new FileInputStream(file);
 byte [] bin_data = new byte [fis.available()];
 fis.read(bin_data);
 String s_bin_data = new String(bin_data);
 st_token = new StringTokenizer(s_bin_data,";");
 fis.close();

} catch (FileNotFoundException e)  {System.out.println(e);}
  catch (java.io.IOException e1)  {System.out.println(e1);};

String server_ip = st_token.nextToken().trim();
String sid = st_token.nextToken().trim();
String uid = st_token.nextToken().trim();
String pass = st_token.nextToken().trim();
String query = st_token.nextToken().trim();

 long t1=0,t2=0,t3=0;

 t1 = (new Date()).getTime();

 try {
  Class.forName( "postgresql.Driver" );
 } catch (ClassNotFoundException e)  {System.out.println(e);}

 java.util.Properties prop = new java.util.Properties();
 String url = "jdbc:postgresql://"+server_ip+"/"+sid+"#EUCJIS";
 prop.put("user", uid);
 prop.put("password", pass);
 prop.put("charset", "8859_1");
// prop.put("charset", "EUCJIS");


 try {

  Connection conn = DriverManager.getConnection(url,prop);

   t2 = (new Date()).getTime();


  Statement st = conn.createStatement();
```

```
//  ResultSet rs = st.executeQuery( "SELECT yubin_no , kanji_cho FROM yubin_ss");
//  ResultSet rs = st.executeQuery( "SELECT yubin_no , kanji_cho FROM yubin_ss where
yubin_no = '6038487' ");
//  ResultSet rs = st.executeQuery( "SELECT yubin_no , kanji_cho FROM yubin_ss where
yubin_no like '%603%' ");
//  String s_sql = "SELECT yubin_no , kanji_cho FROM yubin_ss where kanji_cho like '%
上賀茂%' ";


  ResultSet rs = st.executeQuery(query);

  while (rs.next())  {
   try {
   String s1 = new String(rs.getBytes(1),"EUCJIS");
   String s2 = new String(rs.getBytes(2),"EUCJIS");
   System.out.println("" + s1 + "  **** " + s1);
   } catch (java.io.UnsupportedEncodingException e1) {System.out.println(e1);}
  }
  st.close();
  rs.close();
  conn.close();
} catch (java.sql.SQLException e)  {System.out.println(e);}

 t3 = (new Date()).getTime();

 System.out.println("*** query                    "+query);
 System.out.println("*** dt(connection)  "+(t2-t1)+" (msec)");
 System.out.println("*** dt(query)       "+(t3-t2)+" (msec)");
 System.out.println("*** dt(all)         "+(t3-t1)+" (msec)");

}



    public static void main ( String[] argc)  {
          new test();
    }

  }
```

別添１．４．**実クライアントでのベンチマークテスト用** Java **プログラム（**PostgreSQL **版・更新系）**

```
import java.io.*;
import java.util.*;

import java.sql.*;
import java.util.Date;

public class test {

 public test() {

StringTokenizer st_token =null;
File file = new File("./sg.txt");
try {

  FileInputStream fis = new FileInputStream(file);
 byte [] bin_data = new byte [fis.available()];
 fis.read(bin_data);
 String s_bin_data = new String(bin_data);
 st_token = new StringTokenizer(s_bin_data,";");
 fis.close();

} catch (FileNotFoundException e)  {System.out.println(e);}
  catch (java.io.IOException e1)  {System.out.println(e1);};

String server_ip = st_token.nextToken().trim();
String sid = st_token.nextToken().trim();
String uid = st_token.nextToken().trim();
String pass = st_token.nextToken().trim();
String query = st_token.nextToken().trim();

 long t1=0,t2=0,t3=0;

 t1 = (new Date()).getTime();

 try {
  Class.forName( "postgresql.Driver" );
 } catch (ClassNotFoundException e)  {System.out.println(e);}

 java.util.Properties prop = new java.util.Properties();
 String url = "jdbc:postgresql://"+server_ip+"/"+sid+"#EUCJIS";
 prop.put("user", uid);
 prop.put("password", pass);
 prop.put("charset", "8859_1");
// prop.put("charset", "EUCJIS");


 try {

  Connection conn = DriverManager.getConnection(url,prop);

   t2 = (new Date()).getTime();


  Statement st = conn.createStatement();
```

```
//  ResultSet rs = st.executeQuery( "SELECT yubin_no , kanji_cho FROM yubin_ss");
//  ResultSet rs = st.executeQuery( "SELECT yubin_no , kanji_cho FROM yubin_ss where
yubin_no = '6038487' ");
//  ResultSet rs = st.executeQuery( "SELECT yubin_no , kanji_cho FROM yubin_ss where
yubin_no like '%603%' ");
//  String s_sql = "SELECT yubin_no , kanji_cho FROM yubin_ss where kanji_cho like '%
上賀茂%' ";


  st.executeQuery(query);

//  while (rs.next())  {
//   try {
//   String s1 = new String(rs.getBytes(1),"EUCJIS");
//   String s2 = new String(rs.getBytes(2),"EUCJIS");
//   System.out.println("" + s1 + "  **** " + s1);
//   } catch (java.io.UnsupportedEncodingException e1) {System.out.println(e1);}
//  }
  st.close();
//  rs.close();
  conn.close();
} catch (java.sql.SQLException e)  {System.out.println(e);}

 t3 = (new Date()).getTime();

 System.out.println("*** query                    "+query);
 System.out.println("*** dt(connection)  "+(t2-t1)+" (msec)");
 System.out.println("*** dt(query)       "+(t3-t2)+" (msec)");
 System.out.println("*** dt(all)         "+(t3-t1)+" (msec)");

}



    public static void main ( String[] argc)  {
          new test();
    }

  }
```

別添１．５．**実クライアントでのベンチマークテスト用** Java **プログラム（**Oracle8i **版・参照系）**

```
import java.io.*;
import java.util.*;

import java.sql.*;
import java.util.Date;

public class test {

 public test() {

StringTokenizer st_token =null;
File file = new File("./sg.txt");
try {

  FileInputStream fis = new FileInputStream(file);
 byte [] bin_data = new byte [fis.available()];
 fis.read(bin_data);
 String s_bin_data = new String(bin_data);
 st_token = new StringTokenizer(s_bin_data,";");
 fis.close();

} catch (FileNotFoundException e)  {System.out.println(e);}
  catch (java.io.IOException e1)  {System.out.println(e1);};

String server_ip = st_token.nextToken().trim();
String sid = st_token.nextToken().trim();
String uid = st_token.nextToken().trim();
String pass = st_token.nextToken().trim();
String query = st_token.nextToken().trim();

 long t1=0,t2=0,t3=0;

 t1 = (new Date()).getTime();


 try {
  Class.forName( "oracle.jdbc.driver.OracleDriver" );
 } catch (ClassNotFoundException e)  {System.out.println(e);}

 java.util.Properties prop = new java.util.Properties();
 String url = "jdbc:oracle:thin:@"+server_ip+":1521:"+sid;
 prop.put("user", uid);
 prop.put("password", pass);
 prop.put("charset", "8859_1");
// prop.put("charset", "EUCJIS");


 try {

  Connection conn = DriverManager.getConnection(url,prop);

 t2 = (new Date()).getTime();
```

```
  Statement st = conn.createStatement();
//  ResultSet rs = st.executeQuery( "SELECT yubin_no , kanji_cho FROM yubin_table");
//  ResultSet rs = st.executeQuery( "SELECT yubin_no , kanji_cho FROM yubin_table where
yubin_no = '6038487' ");
//  ResultSet rs = st.executeQuery( "SELECT yubin_no , kanji_cho FROM yubin_table where
yubin_no like '%603%' ");
//  String s_sql = "SELECT yubin_no , kanji_cho FROM yubin_table where kanji_cho like '%
上賀茂%' ";

//  try {
//  s_sql = new String(s_sql.getBytes("SJIS"),"8859_1");
//   } catch (java.io.UnsupportedEncodingException e1) {System.out.println(e1);}

  ResultSet rs = st.executeQuery(query);

  while (rs.next())  {
   String s1 = new String(rs.getString(1));
   String s2 = new String(rs.getString(2));
   System.out.println("" + s1 + "  **** " + s1);
  }
  st.close();
  rs.close();
  conn.close();
} catch (java.sql.SQLException e)  {System.out.println(e);}

 t3 = (new Date()).getTime();

 System.out.println("*** query                    "+query);
 System.out.println("*** dt(connection)  "+(t2-t1)+" (msec)");
 System.out.println("*** dt(query)       "+(t3-t2)+" (msec)");
 System.out.println("*** dt(all)         "+(t3-t1)+" (msec)");


}

    public static void main ( String[] argc)  {
          new test();
    }

  }
```

別添１．６．**実クライアントでのベンチマークテスト用** Java **プログラム（**Oracle8i **版・更新系）**

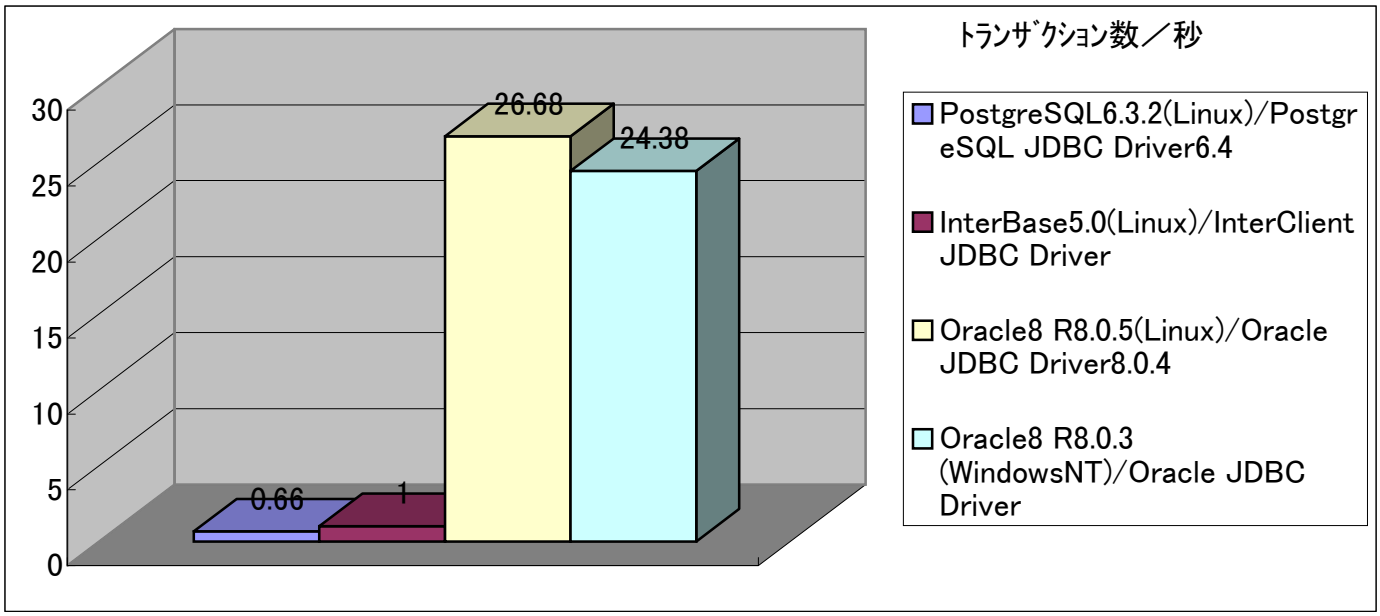
```
import java.io.*;
import java.util.*;

import java.sql.*;
import java.util.Date;

public class test {

 public test() {

StringTokenizer st_token =null;
File file = new File("./sg.txt");
try {

  FileInputStream fis = new FileInputStream(file);
 byte [] bin_data = new byte [fis.available()];
 fis.read(bin_data);
 String s_bin_data = new String(bin_data);
 st_token = new StringTokenizer(s_bin_data,";");
 fis.close();

} catch (FileNotFoundException e)  {System.out.println(e);}
  catch (java.io.IOException e1)  {System.out.println(e1);};

String server_ip = st_token.nextToken().trim();
String sid = st_token.nextToken().trim();
String uid = st_token.nextToken().trim();
String pass = st_token.nextToken().trim();
String query = st_token.nextToken().trim();

 long t1=0,t2=0,t3=0;

 t1 = (new Date()).getTime();


 try {
  Class.forName( "oracle.jdbc.driver.OracleDriver" );
 } catch (ClassNotFoundException e)  {System.out.println(e);}

 java.util.Properties prop = new java.util.Properties();
 String url = "jdbc:oracle:thin:@"+server_ip+":1521:"+sid;
 prop.put("user", uid);
 prop.put("password", pass);
 prop.put("charset", "8859_1");
// prop.put("charset", "EUCJIS");


 try {

  Connection conn = DriverManager.getConnection(url,prop);

 t2 = (new Date()).getTime();
```

```
  Statement st = conn.createStatement();
//  ResultSet rs = st.executeQuery( "SELECT yubin_no , kanji_cho FROM yubin_table");
//  ResultSet rs = st.executeQuery( "SELECT yubin_no , kanji_cho FROM yubin_table where
yubin_no = '6038487' ");
//  ResultSet rs = st.executeQuery( "SELECT yubin_no , kanji_cho FROM yubin_table where
yubin_no like '%603%' ");
//  String s_sql = "SELECT yubin_no , kanji_cho FROM yubin_table where kanji_cho like '%
上賀茂%' ";

//  try {
//  s_sql = new String(s_sql.getBytes("SJIS"),"8859_1");
//   } catch (java.io.UnsupportedEncodingException e1) {System.out.println(e1);}

    st.executeQuery(query);

//  while (rs.next())  {
//   String s1 = new String(rs.getString(1));
//   String s2 = new String(rs.getString(2));
//   System.out.println("" + s1 + "  **** " + s1);
//  }
  st.close();
//  rs.close();
  conn.close();
} catch (java.sql.SQLException e)  {System.out.println(e);}

 t3 = (new Date()).getTime();

 System.out.println("*** query                    "+query);
 System.out.println("*** dt(connection)  "+(t2-t1)+" (msec)");
 System.out.println("*** dt(query)       "+(t3-t2)+" (msec)");
 System.out.println("*** dt(all)         "+(t3-t1)+" (msec)");


}

    public static void main ( String[] argc)  {
          new test();
    }

  }
```

１．JDBCBench による過去のテスト結果

Linux 用 DBMS をオープン・ソース系と商用系を含めて JDBCBench によるベンチマーク結果（吉川和巳（1999）”徹底追跡！Linux 対応データベース最新情報”　Linux WORLD、IDG コミュニケーションズ、P.81、保存版第１弾）を図１．１．に示す。

図１．１．　JDBCBench によるテスト結果



【テスト環境】

●ﾄﾗﾝｻﾞｸｼｮﾝ数

10 ﾄﾗﾝｻﾞｸｼｮﾝ×10 ｸﾗｲｱﾝﾄ（ｽﾚｯﾄﾞ）

●ﾚｺｰﾄﾞ数

branches ﾃｰﾌﾞﾙ：１件

tellers ﾃｰﾌﾞﾙ：10 件

accounts ﾃｰﾌﾞﾙ：10 万件

●ｻｰﾊﾞ環境

200MHz/Pentium 互換 CPU（IDT WinChip）、64MB ﾒﾓﾘ

Linux：Red Hat Linux 5.2 (kernel 2.0.36)

Windows NT：Microsoft Windows NT 4.0 SP4

●ｸﾗｲｱﾝﾄ環境

266MHz/PentiumⅡ、128MB ﾒﾓﾘ

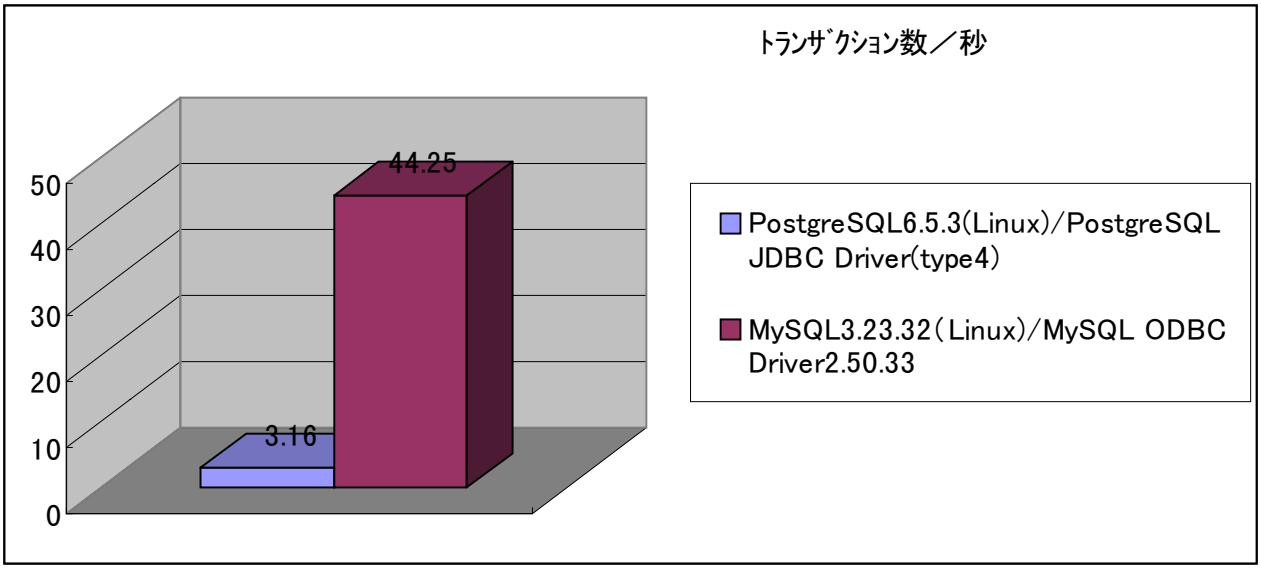
Microsoft Windows NT 4.0 SP4／Sun JDK1.1.7

この結果から判断する限り、PostgreSQL は、商用とは比較にならないくらい、ﾄﾗﾝｻﾞｸｼｮﾝが遅いことがわかる。これは、PostgreSQL がコミットの際、変更されたﾃｰﾌﾞﾙやﾛｸﾞを書き込むたびに fsync を実行していたため、多くのﾄﾗﾝｻﾞｸｼｮﾝを処理するﾍﾞﾝﾁﾏｰｸでは、頻繁にﾃﾞｨｽｸｱｸｾｽが発生し、結果として、ﾊﾟﾌｫｰﾏﾝｽが劣化した、と考えられる。

２．ｵｰﾌﾟﾝｿｰｽ系 DBMS の進化

１.の結果を参考に、最新のﾊﾞｰｼﾞｮﾝのｵｰﾌﾟﾝｿｰｽ系 DBMS を JDBCBench を用いて、ﾍﾞﾝﾁﾏｰｸﾃｽﾄを行った。比較するｵｰﾌﾟﾝｿｰｽ系 DBMS として、PostgreSQL と MySQL を選択した。結果を図２．１．に示す。

図２．１．　JDBCBench によるテスト結果



【テスト環境】

●ﾄﾗﾝｻﾞｸｼｮﾝ数
10 ﾄﾗﾝｻﾞｸｼｮﾝ×10 ｸﾗｲｱﾝﾄ（ｽﾚｯﾄﾞ）
●ﾚｺｰﾄﾞ数
branches ﾃｰﾌﾞﾙ：１件
tellers ﾃｰﾌﾞﾙ：10 件
accounts ﾃｰﾌﾞﾙ：10 万件

●ｻｰﾊﾞ環境
166MHz/Pentium 互換 CPU、
64MB ﾒﾓﾘ
Linux：Red Hat Linux 6.2J (kernel 2.2.14-5.0)
●ｸﾗｲｱﾝﾄ環境
500MHz/Celeron、64MB ﾒﾓﾘ
Microsoft Windows 98／Sun JRE1.2

なお、本論文でも用いているﾍﾞﾝﾁﾏｰｸ用ﾌﾟﾛｸﾞﾗﾑ（JDBCBench.java）中、ﾗﾝﾀﾞﾑにﾃｰﾌﾞﾙを検索できていない箇所があったので、乱数の発生方法・ｽﾚｯﾄﾞの引数を修正している。また、MySQL の JDBC Thin Driver(Type4)は、不具合があるため、JDBC-ODBC ﾌﾞﾘｯｼﾞを使用して接続している。
(MySQL の ODBC ﾄﾞﾗｲﾊﾞ入手先　　http://www.mysql.com/downloads/api-myodbc.html)

結果を見ると、まず、PostgreSQL に関しては、ﾊﾞｰｼﾞｮﾝｱｯﾌﾟに伴い、fsync()によるﾊﾟﾌｫｰﾏﾝｽ劣化が少し解消されたようである。また、MySQL に関しては、ﾄﾗﾝｻﾞｸｼｮﾝはｻﾎﾟｰﾄされないが、商用 DBMS 以上のﾊﾟﾌｫｰﾏﾝｽが期待できそうである。

MySQL、PostgreSQL といったｵｰﾌﾟﾝｿｰｽ系 DBMS では、ﾒｰﾘﾝｸﾞﾘｽﾄを利用した世界中の開発者同士のｺﾗﾎﾞﾚｰｼｮﾝにより、現在も進化を続けている。

（参考１）PostgreSQL ﾍﾞﾝﾁﾏｰｸ前の Linux ｻｰﾊﾞ・ﾌﾟﾛｾｽ

```
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 20:30 ?        00:00:05 init
root         2     1  0 20:30 ?        00:00:00 [kflushd]
root         3     1  0 20:30 ?        00:00:00 [kupdate]
root         4     1  0 20:30 ?        00:00:00 [kpiod]
root         5     1  0 20:30 ?        00:00:00 [kswapd]
root         6     1  0 20:30 ?        00:00:00 [mdrecoveryd]
bin        337     1  0 20:31 ?        00:00:00 portmap
root       351     1  0 20:31 ?        00:00:00 /usr/sbin/apmd -p 10 -w 5 -W -s
root       378     1  0 20:31 ?        00:00:00 /usr/sbin/automount --timeout 60
root       431     1  0 20:31 ?        00:00:00 syslogd -m 0
daemon     472     1  0 20:31 ?        00:00:00 /usr/sbin/atd
root       486     1  0 20:31 ?        00:00:00 crond
postgres   574     1  0 20:31 ?        00:00:00 /usr/bin/postmaster -i -S -D/var
root       751     1  0 20:31 tty1     00:00:00 login -- root
root       775   751  0 21:23 tty1     00:00:00 -bash
root       945     1  0 21:31 tty4     00:00:00 /sbin/mingetty tty4
root       946     1  0 21:31 tty5     00:00:00 /sbin/mingetty tty5
root       948     1  0 21:31 tty6     00:00:00 /sbin/mingetty tty6
root       950     1  0 21:31 tty2     00:00:00 /sbin/mingetty tty2
root       952     1  0 21:32 tty3     00:00:00 /sbin/mingetty tty3
root      1010   775  0 22:03 tty1     00:00:00 ps -ef
```

（参考２）MySQL ﾍﾞﾝﾁﾏｰｸ前の Linux ｻｰﾊﾞ・ﾌﾟﾛｾｽ

```
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  1 01:08 ?        00:00:05 init
root         2     1  0 01:08 ?        00:00:00 [kflushd]
root         3     1  0 01:08 ?        00:00:00 [kupdate]
root         4     1  0 01:08 ?        00:00:00 [kpiod]
root         5     1  0 01:08 ?        00:00:00 [kswapd]
root         6     1  0 01:08 ?        00:00:00 [mdrecoveryd]
bin        332     1  0 01:08 ?        00:00:00 portmap
root       346     1  0 01:08 ?        00:00:00 /usr/sbin/apmd -p 10 -w 5 -W -s
root       373     1  0 01:08 ?        00:00:00 /usr/sbin/automount --timeout 60
root       426     1  0 01:08 ?        00:00:00 syslogd -m 0
daemon     467     1  0 01:08 ?        00:00:00 /usr/sbin/atd
root       481     1  0 01:08 ?        00:00:00 crond
root       779     1  0 01:09 tty1     00:00:00 login -- root
root       780     1  0 01:09 tty2     00:00:00 /sbin/mingetty tty2
root       781     1  0 01:09 tty3     00:00:00 /sbin/mingetty tty3
root       782     1  0 01:09 tty4     00:00:00 /sbin/mingetty tty4
root       783     1  0 01:09 tty5     00:00:00 /sbin/mingetty tty5
root       784     1  0 01:09 tty6     00:00:00 /sbin/mingetty tty6
root       787   779  0 01:09 tty1     00:00:00 -bash
root      1003     1  0 01:14 tty1     00:00:00 sh /usr/bin/safe_mysqld --datadi
mysql     1026  1003  0 01:14 tty1     00:00:00 /usr/sbin/mysqld --basedir=/ --d
mysql     1028  1026  0 01:14 tty1     00:00:00 /usr/sbin/mysqld --basedir=/ --d
mysql     1029  1028  0 01:14 tty1     00:00:00 /usr/sbin/mysqld --basedir=/ --d
root      1033   787  0 01:15 tty1     00:00:00 ps -ef
```

（参考３）PostgreSQL ﾍﾞﾝﾁﾏｰｸ時の Windows ｸﾗｲｱﾝﾄにおける起動ﾊﾞｯﾁ

```
cd C:\WINDOWS\ﾃﾞｽｸﾄｯﾌﾟ\postgresql-java
rem javac -classpath $CLASSPATH;postgresql.jar JDBCBench.java
set CLASSPATH=
C:\Progra~1\JavaSoft\JRE\1.2\bin\java -classpath .;postgresql.jar JDBCBench -driver postgresql.Driver -url jdbc:postgresql://athena/yubin_db#EUCJIS -tpc 10 -clients 10 -v -user pg_linux -pass pg_linux
```

（参考４）MySQL ﾍﾞﾝﾁﾏｰｸ時の Windows ｸﾗｲｱﾝﾄにおける起動ﾊﾞｯﾁ

```
cd C:\WINDOWS\ﾃﾞｽｸﾄｯﾌﾟ\mysql-java
rem javac -classpath $CLASSPATH;mm.mysql.jdbc-1.2b.zip JDBCBench.java
rem javac -classpath $CLASSPATH JDBCBench.java
set CLASSPATH=
C:\Progra~1\JavaSoft\JRE\1.2\bin\java -classpath . JDBCBench -driver sun.jdbc.odbc.JdbcOdbcDriver -url jdbc:odbc:yubin_dsn -tpc 10 -clients 10 -v -user linux -pass linux
```

（参考５）PostgreSQL ﾍﾞﾝﾁﾏｰｸ時の JDBCBench.java

```
/*
 *  This is a sample implementation of the Transaction Processing Performance
 *  Council Benchmark B coded in Java and ANSI SQL2.
 */

import java.sql.*;

public class JDBCBench
{

    /* tpc bm b scaling rules */
    public static int tps      =      1;  /* the tps scaling factor: here it is 1 */
    public static int nbranches =      1;  /* number of branches in 1 tps db       */
    public static int ntellers  =     10;  /* number of tellers in  1 tps db       */
    public static int naccounts =  100000;  /* number of accounts in 1 tps db       */
    public static int nhistory  =  864000;  /* number of history recs in 1 tps db   */

    public final static int TELLER  = 0;
    public final static int BRANCH  = 1;
    public final static int ACCOUNT = 2;



    private Connection Conn = null;

    int failed_transactions = 0;
    int transaction_count = 0;
    static int n_clients = 10;
```

```
    static int n_txn_per_client = 10;
    long start_time = 0;

    static boolean verbose = false;

    MemoryWatcherThread MemoryWatcher;

    /* main program,    creates a 1-tps database:  i.e. 1 branch, 10 tellers,...
     *                    runs one TPC BM B transaction
     */

    public static void main(String[] Args)
    {
      String DriverName = "";
      String DBUrl = "";
//  mente-000 yasui(OSL) 2000/11/12
      String  user = "" , pass="";
//  mente-999 yasui(OSL) 2000/11/12

      boolean initialize_dataset = false;

      for (int i = 0; i < Args.length; i++) {
        if (Args[i].equals("-clients")) {
            if (i + 1 < Args.length) {
                i++;
                n_clients = Integer.parseInt(Args[i]);
            }

        }
        else if (Args[i].equals("-driver")) {
            if (i + 1 < Args.length) {
                i++;
                DriverName = Args[i];
            }
        }
        else if (Args[i].equals("-url")) {
            if (i + 1 < Args.length) {
                i++;
                DBUrl = Args[i];
            }
        }

// mente-000 yasui(OSL) 2000/11/12
        else if (Args[i].equals("-user")) {
            if (i + 1 < Args.length) {
                i++;
                user = Args[i];
            }
        }
        else if (Args[i].equals("-pass")) {
            if (i + 1 < Args.length) {
                i++;
                pass = Args[i];
            }
        }
// mente-999 yasui(OSL) 2000/11/12
```

```
        else if (Args[i].equals("-tpc")) {
            if (i + 1 < Args.length) {
                i++;
                n_txn_per_client = Integer.parseInt(Args[i]);
            }
        }
        else if (Args[i].equals("-init")) {
            initialize_dataset = true;
        }
        else if (Args[i].equals("-v")) {
            verbose = true;
        }
      }

      if (DriverName.length() == 0 || DBUrl.length() == 0) {
        System.out.println("usage: java JDBCBench -driver [driver_class_name] -url [url_to_db] [-v]
[-init] [-tpc n] [-clients]");
         System.out.println();
         System.out.println("-v           verbose error messages");
         System.out.println("-init        initialize the tables");
         System.out.println("-tpc transactions per client");
         System.out.println("-clients    number of simultaneous clients");
        System.exit(-1);
      }

      System.out.println("************************************************************");
      System.out.println("* JDBCBench v1.0                                           *");
      System.out.println("************************************************************");
      System.out.println();
      System.out.println("Driver: " + DriverName);
      System.out.println("URL:" + DBUrl);
      System.out.println();
      System.out.println("Number of clients: " + n_clients);
      System.out.println("Number of transactions per client: " + n_txn_per_client);
      System.out.println();

      try {
        Class.forName(DriverName);
// mente-000 yasui(OSL) 2000/11/12
 //       Connection C = DriverManager.getConnection(DBUrl);
         Connection C = DriverManager.getConnection(DBUrl,user,pass);
// mente-999 yasui(OSL) 2000/11/12

        JDBCBench Me = new JDBCBench(C, initialize_dataset);
      }
      catch (Exception E) {
        System.out.println(E.getMessage());
        E.printStackTrace();
      }
    }

    public JDBCBench(Connection C, boolean init)
    {
        try {
            Conn = C;
```

```
            if (init) {
                System.out.print("Initializing dataset...");
                createDatabase();
                System.out.println("done.\n");
            }
            System.out.println("* Starting Benchmark Run *");
            MemoryWatcher = new MemoryWatcherThread();
            MemoryWatcher.start();

            start_time = System.currentTimeMillis();

            for (int i = 0; i < n_clients; i++) {
                Thread Client = new ClientThread(n_txn_per_client);
                Client.start();
            }
        }
        catch (Exception E) {
            System.out.println(E.getMessage());
            E.printStackTrace();
        }
    }

    public void reportDone()
    {
        n_clients--;

        if (n_clients <= 0) {
            MemoryWatcher.stop();

            long end_time = System.currentTimeMillis();
            double completion_time = ((double)end_time - (double)start_time) / 1000;
            System.out.println("* Benchmark finished *");
            System.out.println("\n* Benchmark Report *" );
            System.out.println("--------------------\n");
            System.out.println("Time to execute " +  transaction_count + " transactions: "  +
completion_time + " seconds.");
            System.out.println("Max/Min memory usage: " + MemoryWatcher.max + " / " +
MemoryWatcher.min + " kb");
            System.out.println(failed_transactions + " / " + transaction_count + " failed to
complete.");
            System.out.println("Transaction rate: " + (transaction_count - failed_transactions) /
completion_time + " txn/sec.");
        }

    }

    public synchronized void incrementTransactionCount()
    {
        transaction_count++;
    }

    public synchronized void incrementFailedTransactionCount()
    {
        failed_transactions++;
    }
```

```
/*
 *  createDatabase() - Creates and Initializes a scaled database.
 */

void createDatabase() throws Exception
{

    try {
        Statement Stmt = Conn.createStatement();


        String Query = "CREATE TABLE branches (";
           Query+= "Bid         INT NOT NULL, PRIMARY KEY(Bid), ";
           Query+= "Bbalance    INT,";
           Query+= "filler      CHAR(88))"; /* pad to 100 bytes */
        Stmt.executeUpdate(Query);
        Stmt.clearWarnings();

        Query = "CREATE TABLE tellers ( ";
        Query+= "Tid         INT NOT NULL, PRIMARY KEY(Tid),";
        Query+= "Bid         INT,";
        Query+= "Tbalance    INT,";
        Query+= "filler      CHAR(84))"; /* pad to 100 bytes */

        Stmt.executeUpdate(Query);
        Stmt.clearWarnings();

        Query = "CREATE TABLE accounts ( ";
        Query+= "Aid         INT NOT NULL, PRIMARY KEY(Aid), ";
        Query+= "Bid         INT, ";
        Query+= "Abalance    INT, ";
        Query+= "filler      CHAR(84))";      /* pad to 100 bytes */

        Stmt.executeUpdate(Query);
        Stmt.clearWarnings();

        Query = "CREATE TABLE history ( ";
        Query+= "Tid         INT, ";
        Query+= "Bid         INT, ";
        Query+= "Aid         INT, ";
        Query+= "delta       INT, ";
        Query+= "time        TIMESTAMP, ";
        Query+= "filler      CHAR(22))"; /* pad to 50 bytes  */

        Stmt.executeUpdate(Query);
        Stmt.clearWarnings();

        /* prime database using TPC BM B scaling rules.
         *  Note that for each branch and teller:
         *      branch_id = teller_id  / ntellers
         *      branch_id = account_id / naccounts
         */

        for (int i = 0; i < nbranches * tps; i++) {
            Query = "INSERT INTO branches(Bid,Bbalance) VALUES (" + i + ",0)";
```

```
                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();
            }
            for (int i = 0; i < ntellers * tps; i++) {
                Query = "INSERT INTO tellers(Tid,Bid,Tbalance) VALUES (" + i + "," + i / ntellers
+ ",0)";
                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();
            }
            for (int i = 0; i < naccounts*tps; i++) {
                Query = "INSERT INTO accounts(Aid,Bid,Abalance) VALUES (" + i + "," + i / naccounts
+ ",0)";
                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();
            }
// mente-000 yasui(OSL) 2000/11/12
            Stmt.close();
// mente-999 yasui(OSL) 2000/11/12

        }
        catch (Exception E) {
            System.out.println(E.getMessage());
            E.printStackTrace();
        }

    }           /* end of CreateDatabase    */

    public static int getRandomInt(int lo, int hi)
    {
          int ret = 0;

          ret = (int)(Math.random() * (hi - lo + 1));
          ret += lo;

          return ret;
    }

    public static int getRandomID(int type)
    {
        int min, max, num;

        max = min = 0;
        num = naccounts;

        switch(type) {
        case TELLER:
                min += nbranches;
                num = ntellers;
                /* FALLTHROUGH */
        case BRANCH:
                if (type == BRANCH)
                        num = nbranches;
                min += naccounts;
                /* FALLTHROUGH */
        case ACCOUNT:
                max = min + num - 1;
```

```
        }
        return (getRandomInt(min, max));
    }

    class ClientThread extends Thread
    {
        int ntrans = 0;

        public ClientThread(int number_of_txns)
        {
            ntrans = number_of_txns;
        }

        public void run()
        {
            while (ntrans-- > 0) {
  /*
                int account = JDBCBench.getRandomID(ACCOUNT);
                int branch  = JDBCBench.getRandomID(BRANCH);
                int teller  = JDBCBench.getRandomID(TELLER);
*/
                int account = JDBCBench.getRandomInt(0,naccounts-1);
                int branch  = JDBCBench.getRandomInt(0,nbranches-1);
                int teller  = JDBCBench.getRandomInt(0,ntellers-1);

                int delta   = JDBCBench.getRandomInt(0,1000);

                doOne(branch, teller, account, delta);
                incrementTransactionCount();
            }
            reportDone();
        }

        /*
         *  doOne() - Executes a single TPC BM B transaction.
         */

        int doOne(int bid, int tid, int aid, int delta)
        {
            try {
                Statement Stmt = Conn.createStatement();

                String Query = "UPDATE accounts ";
                       Query+= "SET     Abalance = Abalance + " + delta + " ";
                       Query+= "WHERE   Aid = " + aid;

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

                Query = "SELECT Abalance ";
                Query+= "FROM   accounts ";
                Query+= "WHERE  Aid = " + aid;

                ResultSet RS = Stmt.executeQuery(Query);
                Stmt.clearWarnings();
```

```
                int aBalance = 0;

                while (RS.next()) {
                    aBalance = RS.getInt(1);
                }

                Query = "UPDATE tellers ";
                Query+= "SET    Tbalance = Tbalance + " + delta + " ";
                Query+= "WHERE  Tid = " + tid;

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

                Query = "UPDATE branches ";
                Query+= "SET    Bbalance = Bbalance + " + delta + " ";
                Query+= "WHERE  Bid = " + bid;

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

                Query = "INSERT INTO history(Tid, Bid, Aid, delta) ";
                Query+= "VALUES (";
                Query+= tid + ",";
                Query+= bid + ",";
                Query+= aid + ",";
                Query+= delta + ")";

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

// mente-000 yasui(OSL) 2000/11/12
            Stmt.close();
                     RS.close();
// mente-999 yasui(OSL) 2000/11/12

                return aBalance;
            }
            catch (SQLException E) {
                if (verbose) {
                    System.out.println("Transaction failed: " + E.getMessage());
                    E.printStackTrace();
                }
                incrementFailedTransactionCount();
            }
            return 0;

        }           /* end of DoOne         */




    }

    class MemoryWatcherThread extends Thread
    {
        long min = 0;
```

```
        long max = 0;

        public void run()
        {
            min = Runtime.getRuntime().freeMemory();

            for (;;) {
                long currentFree = Runtime.getRuntime().freeMemory();
                long currentAlloc = Runtime.getRuntime().totalMemory();
                long used = currentAlloc - currentFree;

                if (used < min)
                    min = used;
                if (used > max)
                    max = used;

                try {
                    sleep(100);
                }
                catch (InterruptedException E) {}
            }
         }
    }


}
```

（参考６）MySQL ﾍﾞﾝﾁﾏｰｸ時の JDBCBench.java

```
/*
 *  This is a sample implementation of the Transaction Processing Performance
 *  Council Benchmark B coded in Java and ANSI SQL2.
 */

import java.sql.*;

public class JDBCBench
{
```

```
    /* tpc bm b scaling rules */
    public static int tps       =      1;  /* the tps scaling factor: here it is 1 */
    public static int nbranches =      1;  /* number of branches in 1 tps db       */
    public static int ntellers  =     10;  /* number of tellers in  1 tps db       */
    public static int naccounts =  100000;  /* number of accounts in 1 tps db       */
    public static int nhistory  =  864000;  /* number of history recs in 1 tps db   */

    public final static int TELLER  = 0;
    public final static int BRANCH  = 1;
    public final static int ACCOUNT = 2;



    private Connection Conn = null;

    int failed_transactions = 0;
    int transaction_count = 0;
    static int n_clients = 10;
    static int n_txn_per_client = 10;
    long start_time = 0;

    static boolean verbose = false;

    MemoryWatcherThread MemoryWatcher;

    /* main program,    creates a 1-tps database:  i.e. 1 branch, 10 tellers,...
     *                    runs one TPC BM B transaction
     */

    public static void main(String[] Args)
    {
      String DriverName = "";
      String DBUrl = "";
//  mente-000 yasui(OSL) 2000/11/12
      String  user = "" , pass="";
//  mente-999 yasui(OSL) 2000/11/12

      boolean initialize_dataset = false;

      for (int i = 0; i < Args.length; i++) {
        if (Args[i].equals("-clients")) {
            if (i + 1 < Args.length) {
                i++;
                n_clients = Integer.parseInt(Args[i]);
            }

        }
        else if (Args[i].equals("-driver")) {
            if (i + 1 < Args.length) {
                i++;
                DriverName = Args[i];
            }
        }
        else if (Args[i].equals("-url")) {
            if (i + 1 < Args.length) {
                i++;
```

```
                DBUrl = Args[i];
            }
        }

// mente-000 yasui(OSL) 2000/11/12
        else if (Args[i].equals("-user")) {
            if (i + 1 < Args.length) {
                i++;
                user = Args[i];
            }
        }
        else if (Args[i].equals("-pass")) {
            if (i + 1 < Args.length) {
                i++;
                pass = Args[i];
            }
        }
// mente-999 yasui(OSL) 2000/11/12

        else if (Args[i].equals("-tpc")) {
            if (i + 1 < Args.length) {
                i++;
                n_txn_per_client = Integer.parseInt(Args[i]);
            }
        }
        else if (Args[i].equals("-init")) {
            initialize_dataset = true;
        }
        else if (Args[i].equals("-v")) {
            verbose = true;
        }
      }

      if (DriverName.length() == 0 || DBUrl.length() == 0) {
        System.out.println("usage: java JDBCBench -driver [driver_class_name] -url [url_to_db] [-v]
[-init] [-tpc n] [-clients]");
         System.out.println();
         System.out.println("-v           verbose error messages");
         System.out.println("-init        initialize the tables");
         System.out.println("-tpc transactions per client");
         System.out.println("-clients    number of simultaneous clients");
        System.exit(-1);
      }

      System.out.println("*************************************************************");
      System.out.println("* JDBCBench v1.0                                            *");
      System.out.println("*************************************************************");
      System.out.println();
      System.out.println("Driver: " + DriverName);
      System.out.println("URL:" + DBUrl);
      System.out.println();
      System.out.println("Number of clients: " + n_clients);
      System.out.println("Number of transactions per client: " + n_txn_per_client);
      System.out.println();

      try {
```

```
        Class.forName(DriverName);


// mente-000 yasui(OSL) 2000/11/12
//       Connection C = DriverManager.getConnection(DBUrl,"","");
         Connection C = DriverManager.getConnection(DBUrl,user,pass);
// mente-999 yasui(OSL) 2000/11/12

        JDBCBench Me = new JDBCBench(C, initialize_dataset);
      }
      catch (Exception E) {
        System.out.println(E.getMessage());
        E.printStackTrace();
      }
    }

    public JDBCBench(Connection C, boolean init)
    {
        try {
            Conn = C;
            if (init) {
                System.out.print("Initializing dataset...");
                createDatabase();
                System.out.println("done.\n");
            }
            System.out.println("* Starting Benchmark Run *");
            MemoryWatcher = new MemoryWatcherThread();
            MemoryWatcher.start();

            start_time = System.currentTimeMillis();

            for (int i = 0; i < n_clients; i++) {
                Thread Client = new ClientThread(n_txn_per_client);
                Client.start();
            }
        }
        catch (Exception E) {
            System.out.println(E.getMessage());
            E.printStackTrace();
        }
    }

    public void reportDone()
    {
        n_clients--;

        if (n_clients <= 0) {
            MemoryWatcher.stop();

            long end_time = System.currentTimeMillis();
            double completion_time = ((double)end_time - (double)start_time) / 1000;
            System.out.println("* Benchmark finished *");
            System.out.println("\n* Benchmark Report *" );
            System.out.println("--------------------\n");
            System.out.println("Time to execute " +  transaction_count + " transactions: "  +
completion_time + " seconds.");
```

```
            System.out.println("Max/Min memory usage: " + MemoryWatcher.max + " / " +
MemoryWatcher.min + " kb");
            System.out.println(failed_transactions + " / " + transaction_count + " failed to
complete.");
            System.out.println("Transaction rate: " + (transaction_count - failed_transactions) /
completion_time + " txn/sec.");
        }

    }

    public synchronized void incrementTransactionCount()
    {
        transaction_count++;
    }

    public synchronized void incrementFailedTransactionCount()
    {
        failed_transactions++;
    }


    /*
     *  createDatabase() - Creates and Initializes a scaled database.
     */

    void createDatabase() throws Exception
    {

        try {
            Statement Stmt = Conn.createStatement();


            String Query = "CREATE TABLE branches (";
               Query+= "Bid         INT NOT NULL, PRIMARY KEY(Bid), ";
               Query+= "Bbalance    INT,";
               Query+= "filler      CHAR(88))"; /* pad to 100 bytes */
            Stmt.executeUpdate(Query);
            Stmt.clearWarnings();

            Query = "CREATE TABLE tellers ( ";
            Query+= "Tid         INT NOT NULL, PRIMARY KEY(Tid),";
            Query+= "Bid         INT,";
            Query+= "Tbalance    INT,";
            Query+= "filler      CHAR(84))"; /* pad to 100 bytes */

            Stmt.executeUpdate(Query);
            Stmt.clearWarnings();

            Query = "CREATE TABLE accounts ( ";
            Query+= "Aid         INT NOT NULL, PRIMARY KEY(Aid), ";
            Query+= "Bid         INT, ";
            Query+= "Abalance    INT, ";
            Query+= "filler      CHAR(84))";      /* pad to 100 bytes */

            Stmt.executeUpdate(Query);
            Stmt.clearWarnings();
```

```
            Query = "CREATE TABLE history ( ";
            Query+= "Tid         INT, ";
            Query+= "Bid         INT, ";
            Query+= "Aid         INT, ";
            Query+= "delta       INT, ";
            Query+= "time        TIMESTAMP, ";
            Query+= "filler      CHAR(22))"; /* pad to 50 bytes  */

            Stmt.executeUpdate(Query);
            Stmt.clearWarnings();

            /* prime database using TPC BM B scaling rules.
             *  Note that for each branch and teller:
             *      branch_id = teller_id  / ntellers
             *      branch_id = account_id / naccounts
             */

            for (int i = 0; i < nbranches * tps; i++) {
                Query = "INSERT INTO branches(Bid,Bbalance) VALUES (" + i + ",0)";
                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();
            }
            for (int i = 0; i < ntellers * tps; i++) {
                Query = "INSERT INTO tellers(Tid,Bid,Tbalance) VALUES (" + i + "," + i / ntellers
+ ",0)";
                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();
            }
            for (int i = 0; i < naccounts*tps; i++) {
                Query = "INSERT INTO accounts(Aid,Bid,Abalance) VALUES (" + i + "," + i / naccounts
+ ",0)";
                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();
            }
// mente-000 yasui(OSL) 2000/11/12
            Stmt.close();
// mente-999 yasui(OSL) 2000/11/12

        }
        catch (Exception E) {
            System.out.println(E.getMessage());
            E.printStackTrace();
        }

    }           /* end of CreateDatabase    */

    public static int getRandomInt(int lo, int hi)
    {
          int ret = 0;

          ret = (int)(Math.random() * (hi - lo + 1));
          ret += lo;

          return ret;
    }
```

```
    public static int getRandomID(int type)
    {
        int min, max, num;

        max = min = 0;
        num = naccounts;

        switch(type) {
        case TELLER:
                min += nbranches;
                num = ntellers;
                /* FALLTHROUGH */
        case BRANCH:
                if (type == BRANCH)
                        num = nbranches;
                min += naccounts;
                /* FALLTHROUGH */
        case ACCOUNT:
                max = min + num - 1;
        }
        return (getRandomInt(min, max));
    }

    class ClientThread extends Thread
    {
        int ntrans = 0;

        public ClientThread(int number_of_txns)
        {
            ntrans = number_of_txns;
        }

        public void run()
        {
            while (ntrans-- > 0) {
  /*
                int account = JDBCBench.getRandomID(ACCOUNT);
                int branch  = JDBCBench.getRandomID(BRANCH);
                int teller  = JDBCBench.getRandomID(TELLER);
*/
                int account = JDBCBench.getRandomInt(0,naccounts-1);
                int branch  = JDBCBench.getRandomInt(0,nbranches-1);
                int teller  = JDBCBench.getRandomInt(0,ntellers-1);

                int delta   = JDBCBench.getRandomInt(0,1000);

                doOne(branch, teller, account, delta);
                incrementTransactionCount();
            }
            reportDone();
        }

        /*
         *  doOne() - Executes a single TPC BM B transaction.
         */
```

```
        int doOne(int bid, int tid, int aid, int delta)
        {
            try {
                Statement Stmt = Conn.createStatement();

                String Query = "UPDATE accounts ";
                       Query+= "SET     Abalance = Abalance + " + delta + " ";
                       Query+= "WHERE   Aid = " + aid;

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

                Query = "SELECT Abalance ";
                Query+= "FROM   accounts ";
                Query+= "WHERE  Aid = " + aid;

                ResultSet RS = Stmt.executeQuery(Query);
                Stmt.clearWarnings();

                int aBalance = 0;

                while (RS.next()) {
                    aBalance = RS.getInt(1);
                }

// mente-000 yasui(OSL) 2001/2/3
//                if (!(RS == null)) {
                            RS.close();
 //                         }
// mente-999 yasui(OSL) 2001/2/3
                Query = "UPDATE tellers ";
                Query+= "SET    Tbalance = Tbalance + " + delta + " ";
                Query+= "WHERE  Tid = " + tid;

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

                Query = "UPDATE branches ";
                Query+= "SET    Bbalance = Bbalance + " + delta + " ";
                Query+= "WHERE  Bid = " + bid;

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

                Query = "INSERT INTO history(Tid, Bid, Aid, delta) ";
                Query+= "VALUES (";
                Query+= tid + ",";
                Query+= bid + ",";
                Query+= aid + ",";
                Query+= delta + ")";

                Stmt.executeUpdate(Query);
                Stmt.clearWarnings();

// mente-000 yasui(OSL) 2000/11/12
```

```
                Stmt.close();
// mente-999 yasui(OSL) 2000/11/12
// System.exit(0);
                return aBalance;
            }
            catch (SQLException E) {


                if (verbose) {
                    System.out.println("Transaction failed: " + E.getMessage());
                    E.printStackTrace();
                }
                incrementFailedTransactionCount();
            }
            return 0;

        }           /* end of DoOne         */




    }

    class MemoryWatcherThread extends Thread
    {
        long min = 0;
        long max = 0;

        public void run()
        {
            min = Runtime.getRuntime().freeMemory();

            for (;;) {
                long currentFree = Runtime.getRuntime().freeMemory();
                long currentAlloc = Runtime.getRuntime().totalMemory();
                long used = currentAlloc - currentFree;

                if (used < min)
                    min = used;
                if (used > max)
                    max = used;

                try {
                    sleep(100);
                }
                catch (InterruptedException E) {}
            }
         }
    }



}
```