Redbooks Paper

John Gray
Lydia Parziale

Tuning Considerations: PeopleSoft North American Payroll V8.8 for DB2 V7.1

zSeries® customers are always looking for ways to improve the performance of their business systems. With a few tuning tips from this paper, they can achieve remarkable performance results.

This IBM® Redpaper contains some real world experiences involving initial tuning considerations of PeopleSoft North American Payroll Version 8.8 for DB2® Version 7.1 under the IBM z/OS™ 1.2 operating environment.

Introduction

In the spring of 2003, John Gray of PeopleSoft set out to benchmark the PeopleSoft Payroll V8.8 (North American) application under Version 7.1 of IBM DB2 software. The goal of the project was to exercise the newest versions of PeopleSoft and DB2 code using IBM new Z990 processor and latest Enterprise Storage System (ESS) hardware under the z/OS operating system, to show customers the results that can be achieved by using these recent advances in enterprise platform technology.

The objective was not to achieve an ultimate state of tuning in a vacuum environment, but rather to represent a moderately tuned application run under real-world conditions. The results reflected in this benchmark should therefore be representative of what would be achievable by any production shop operating under a similar environment.

John wrote the white paper that is the basis of this paper. He was assisted by Mike Curtis of the IBM zSeries International Competency Center during the benchmark, in the areas of system monitoring and performance analysis.

Business process background

The three PeopleSoft Payroll processes tested are as follows:

- Paysheet Creation

  Generates payroll data worksheets for employees, consisting of standard payroll information for each employee for the given pay cycle. The Paysheet process can be run separately from the other two tasks, usually before the end of the pay period.

- Payroll Calculation

  Looks at Paysheets and calculates checks for those employees. Payroll Calculation can be run any number of times throughout the pay period. The first run will do most of the processing, while each successive run updates only the calculated totals of changed items. This iterative design minimizes the time required to calculate a payroll, as well as the processing resources required. In this benchmark, Payroll Calculation was run only once, as though at the end of a pay period.

- Payroll Confirmation

  Takes the information generated by Payroll Calculation and updates the employees' balances with the calculated amounts. The system assigns check numbers at this time and creates direct deposit records. Confirm can only be run once, and therefore, must be run at the end of the pay period.

Environment profile

The environment used in earlier benchmark tests performed at PeopleSoft in Pleasanton, CA, consisted of running PeopleSoft Payroll V8.8 (North American) using IBM® DB2 for OS/390®™ 7.1 on an IBM® zSeries 900 model 2064-114 database server, running IBM® z/OS Version 1.2. The Z900 Model 2064-114 had 14 engines, of which, 4 were utilized for the purposes of the test. The Z900 unit contained volumes totaling 2.4 TB of storage. 1.2 TB was available in the SMS-managed pool.

The environment used in this project consists of a DB2 subsystem that has the ability to access shared DB2 data. This data sharing group has a single data sharing member and runs on a collection of MVS™ systems (a parallel sysplex). The single data sharing member is running z/OS Version 1.2 on an IBM Z990 enterprise server. Specifically, the Z990 used was a 2084-B16 with model feature 313. This indicates 13 engines, of which, 6 were utilized for purposes of this test. This configuration has 64 GB of memory total; 6 GB of memory was available to the DB2 subsystem used in this test. Data was stored on a new IBM ESS (Enterprise Storage System) Shark drive; Model 2105-800 Turbo with 64 GB cache. This unit contains volumes totaling 7 TB of storage. 4 TB was available in the SMS managed pool used in this test and payroll data occupied approximately 300 GB. All data was defined in a single SMS-managed storage pool. The Parallel Access Volume (PAV) feature was enabled.

Workload

The workload in this exercise was derived from a standard Large Model PeopleSoft Payroll V8.8 benchmark toolkit. Sixteen paygroups were used to provide the data used in this test, consisting of 5,630 employees in each paygroup, for a total of 90,080 employees. 72,064 employees are active—but one of the employees gets two checks, so a total of 90,080 checks are run. Sixteen processes were run in parallel for each of the three payroll processing steps: Paysheet Creation, Payroll Calculation, and Payroll Confirmation.

Payroll benchmark details

Three North American Payroll benchmarks were run using PeopleSoft's DB2 environment. These benchmarks measured performance results for three key business processes: Paysheet creation, payroll calculation and payroll confirmation. The tests were designed to provide an accurate picture of the system's performance in a production setting. The system tests were performed first on the z900 platform and then on the z990 platform. The environments are defined in "Environment profile" on page 2.

The following combinations of software and platform were used:

- N.A. Payroll Version 8.1 on a Z900 platform
- N.A. Payroll Version 8.8 on the same Z900 platform
- N.A. Payroll Version 8.8 on the new Z990 platform

All three tests were conducted using the same 8.42 version of PeopleTools.

Table 1 contains the actual runtimes, in minutes, for the Payroll processes. It also shows the total runtime as well as the total checks per hour.

Table 1 Results of payroll processing benchmarks

	Z900 Process Standard Large Model (90,000 employees)		**Z990 Process Standard Large Model (90,000 employees)**
	PeopleSoft Payroll V8.1	PeopleSoft Payroll V8.8	PeopleSoft Payroll V8.8
Paysheet Creation	2.01	1.74	.67
Payroll Calculation	28.55	13.52	7.04
Payroll Confirmation	7.66	11.50	4.25
Total	38.22	26.76	11.96
Checks/hour	142,105	201,793	451,505

Tuning considerations

This section details some of the environmental settings and steps that were taken to achieve efficient processing. The general parameters around this test were to achieve results that could be duplicated in the type of real-world environment found in the average production shop. Therefore, this test attempts to show the kind of results that can be expected from this application when bufferpool sizings and other system parameters are tailored to a moderate extent specifically for a DB2 environment.

These results were obtained solely by implementing the tuning procedures described below.

The changes described herein are only an example of this type of environmental tuning and setup and are not meant to imply that they would be correct for all DB2 environments, or that they should be implemented as is. Each enterprise environment is complex and proper controls should be used when introducing changes to your environment.

Tablespace sizing

The first issue to be addressed was at the tablespace creation point in the installation process. Tablespace sizes, as found in the PeopleSoft Payroll benchmark toolkit, are set very small by default. An OS/390 or z/OS environment requires a precise attempt at tablespace sizing to both successfully load the data and efficiently process it. In fact, many of the tablespaces will not load “large model” volumes of data with the default sizes found in the toolkit in an MVS environment. A maximum of 251 extents will be given for the underlying VSAM datasets; after that the load process will abend.

Even if a given larger table *can* be loaded with the default primary and secondary values, quite often it will have been created using a large number of extents and performance can be adversely affected. According to IBM, this ranges from less of an issue to a total non-issue with the new ESS Shark family of storage devices, but a minimum number of extents was still adhered to in this test to eliminate the possibility. The rule of thumb used in this tuning effort was to keep the number of extents taken by the system under 30. The sizings that we used, as shown in “Appendix A. Tablespace sizing” on page 17, are large enough to guarantee this extent range, but are not necessarily the most efficient sizings that could be determined for use in a production shop.

All tablespaces were created with COMPRESS YES to also aid in efficient disk space use. Additionally, indexes were also resized where necessary for the same reasons. COMPRESS YES on large tables with numerous indexes may not always yield the intended results as it becomes a potential virtual storage issue. For this environment configuration, we had successful results.

A Share.org presentation from 2001 used the example of PS_JOB with six indexes. Since indexes do not compress, the 80 percent perceived savings from compression in this case may only be 15 percent, along with the added overhead of up to 64 k in the DB1M address space for the compression dictionary. Loads, inserts, and updates will also be more CPU expensive. The benefit, aside from any DASD savings, is that more data can be placed in the buffer pools if it is compressed, which improves the buffer pool hit ratio. Large tables should be evaluated individually with factors like online access, number and frequency of queries against the table that return large numbers of rows, system virtual storage constraints, etc., all playing a role in determining if compression should be used.

The exact sizes of each tablespace that we created can be found in “Appendix A. Tablespace sizing” on page 17. The Share presentation can be found at:

```
http://www.share.org/proceedings/sh96/data/S1347.PDF
```

Tablespace tuning

While DB2 data is accessed at the table level, the actual data is stored in tablespaces. In DB2, storage is handled in pages within the tablespace. Depending on the tablespace page size and the size of rows in your tables, a row may occupy many pages, or a page may contain many rows. Storing a row, or many rows, in a single page, allows the entire row or rows to be retrieved as a single unit from disk, thus enhancing performance in data retrieval. This is particularly useful for tables that experience high data access.

A listing of tablespaces with their respective tables can be found in "Appendix B. Table and tablespace mappings" on page 19. In this listing we show the new tablespaces that were created in addition to the ones contained in the PeopleSoft toolkit, and the tables listed next to them were separated into new tablespaces for these performance reasons.

Tablespace lock settings

Following initial testing, the tablespaces in "Appendix C. Tablespace lock settings" on page 21 were set initially from installation, or altered, to row or page level locking in order to improve scan efficiency on their corresponding tables. Tuning locks this way in DB2 reduces, if not eliminates, the likelihood of transaction timeouts and deadlocks. While this requires some overhead to manage, you can achieve better concurrency during data access by tuning your lock settings.

Bufferpool considerations

Tablespaces can be associated with bufferpools at tablespace creation time. Bufferpools are a primary system tuning resource. Using bufferpools effectively will optimize both the application performance and the use of DB2 memory resources.

The standard implementation procedure for installing PeopleSoft applications in an OS/390 or z/OS environment calls for all tablespaces except PSIMAGE and PSIMGR to be created in BP3 and all indexes to be created in BP4. PSIMAGE and PSIMGR require a BP32K bufferpool. To optimize performance, we chose to create specific bufferpools for specific tablespaces.

Table 2 on page 24 shows the bufferpool settings that were in place at the time of each benchmark. As shown in the Payroll benchmark details table, Table 1 on page 3, benchmark results for PeopleSoft Payroll V8.1 on the Z900 show a total processing time of 38.22 minutes. A substantial part of the improvement in total runtime to 26.76 minutes for the PeopleSoft Payroll V8.8 on the Z900 was through greater efficiency and throughput achieved from bufferpool tuning. Bufferpools were increased in size as detailed in Table 2 on page 24, and any bufferpool larger than 10,000 was converted to use dataspaces. The appearance of new bufferpools, such as BP31-33, were created for other benchmarks executing in the same DB2 subsystem and are listed here for documentation purposes only.

Once the Z990 and Shark were installed and the PeopleSoft Payroll V8.8 on the Z990 testing was underway, settings for the subsystem bufferpools were only changed to accommodate another benchmark test that was executing in the same subsystem. These changes did not benefit PeopleSoft Payroll runtimes. Changes listed in this column of Table 2 on page 24 are for documentation purposes only, not as a performance recommendation. The increases were mostly in bufferpools not used by Payroll, that is, BP33. In fact, BP3 and BP4, containing the majority of tables and indexes used by Payroll, actually decreased by 20,000 and 10,000, respectively, from the time of the previous test.

In tuning the PeopleSoft V8.8 on the Z900 benchmark, the following tablespaces were moved to separate bufferpools for performance reasons. These settings were retained for the PeopleSoft V8.8 on the Z990 benchmark:

- Tablespace BNAPP24 was altered to bufferpool BP24 (contains table PS_LIMIT_INCLD_TBL).
- Tablespace BNAPP25 was altered to bufferpool BP25 (contains table PS_LIMIT_IMPIN_TBL).
- Tablespace BNAPP26 was altered to bufferpool BP26 (contains table PS_LIMIT_EXCLD_TBL).
- Tablespace PYLARG5 was altered to bufferpool BP6 (contains table PS_PAY_EARNINGS).
- Tablespace PYLARG1 was altered to bufferpool BP21 (contains table PS_DEDUCTION_BAL).
- Tablespace HRLARG1 was altered to bufferpool BP24 (contains table PS_JOB).
- Tablespace PTPRC was altered to bufferpool BP20 (contains table PSSERVERSTAT among others).
- Indexes for two tables were also moved to their own bufferpools for performance reasons.
- Indexes for PS_JOB were moved to BP7.
- Indexes for PS_PAY_EARNINGS, including the new PSCPAY_EARNINGS, were moved to BP10.

Increases in size for other bufferpools not shown as being used in this Payroll benchmark (that is, BP33) are due to another benchmark executing in the same DB2 subsystem.

Other performance changes

SQL Statement traces were run to isolate and analyze the worst-performing statements in the Payroll processes. Subsequent analysis revealed that the four worst performing statements could all be improved with the addition of one index on the PS_PAY_LINE table. A UDB benchmark on PeopleSoft V8.8 North American Payroll was underway during this time and statement traces were compared. Interestingly, the UDB and DB2 optimizers choose completely different paths for execution of these statements, and all four statements run fine in UDB. This change is therefore specific to the DB2 platform, but results in markedly better execution times as shown below.

PAYCALC

This was the longest running statement in Paycalc: 77.487 seconds out of 200 seconds total for the job. Worse, it returns *zero* rows.

```
0.002     0.002 #001 RC= 100 Execute
0.002     0.000 #001 RC=   0 Row Count=000000000
0.000     0.000 #002 RC= 100 Fetch
0.000     0.000 #002 RC=   0 Close Cursor for PSPSTRUN_S_CAL
0.000     0.000 #002 RC=   0 Bind=0001 Type=SQLPBUF Len=0009 Data=KU
0.000     0.000 #002 RC=   0 Execute
0.000     0.000 #002 RC=   0 Fetch
0.000     0.000      RC=   0 GETSTMT Stmt=PSPSTRUN_U_STA1
6.007     0.007 #001 RC=   0 Prepare=UPDATE PS_PAY_EARNINGS
                             SET PAY_LINE_STATUS='P' WHERE COMPANY=
                             :1 AND PAY_END_DT=:2 AND OFF_CYCLE=:3
                             AND SINGLE_CHECK_USE IN ('N','P')
```

```
                              AND PAY_LINE_STATUS='E'
  0.011     0.000 #001 RC=   0 Bind=0001 Type=SQLPBUF Len=0003 Data=GB
  0.000     0.000 #001 RC=   0 Bind=0002 Type=0384 Len=0010 Data=2000-
  0.000     0.000 #001 RC=   0 Bind=0003 Type=SQLPBUF Len=0001 Data=N
 77.487    77.487 #001 RC= 100 Execute
 77.487     0.000 #001 RC=   0 Row Count=000000000
  0.003     0.000 #001 RC=   0 Commit
  0.000     0.000      RC=   0 GETSTMT Stmt=PSPSTRUN_D_CHK
------------------------------------------------------------------------------
```

The Explain showed:

```
--------+---------+---------+---------+---------+---------+---------+---------+
QRYNO  QBLKNO PLANNO METHOD  TNAME       ACCESSTYPE  MATCHCOLS ACCESSNAME   INDEXONLY
--------+---------+---------+---------+---------+---------+---------+---------+
1011     1      1      0  PS_PAY_EARNINGS     I         0     PSDPAY_EARNINGS  N
1011     1      1      0  PS_PAY_EARNINGS     I         0     PSDPAY_EARNINGS  N
1011     1      1      0  PS_PAY_EARNINGS     I         0     PSDPAY_EARNINGS  N
PSDPAY_EARNINGS  FIRSTKEYCARD  FULLKEYCARD   CLUSTERRATIO
                 -------+---------+---------+-------------
                   73,623       1,040,557        97
===============================================================================
```

The second longest running statement in Paycalc was 74.851 seconds out of 200.

Sixty-ne rows were returned.

```
Prepare=UPDATE PS_PAY_LINE
SET SINGLE_CHECK_USE='N'
,PAY_LINE_STATUS='U'
WHERE COMPANY=:1 AND PAY_END_DT=:2
AND OFF_CYCLE=:3 AND
SINGLE_CHECK_USE='P' AND
PAY_LINE_STATUS <> 'F';
  0.037     0.000 #001 RC=   0 Bind=0001 Type=SQLPBUF Len=0003 Data=G
  0.000     0.000 #001 RC=   0 Bind=0002 Type=SQLPBUF Len=0003 Data=K
  0.000     0.000 #001 RC=   0 Bind=0003 Type=0384 Len=0010 Data=2000
  0.000     0.000 #001 RC=   0 Bind=0004 Type=SQLPBUF Len=0001 Data=N
  0.001     0.001 #001 RC=   0 Execute
  0.001     0.000 #001 RC=   0 Row Count=000000001
  0.000     0.000      RC=   0 GETSTMT Stmt=PSPCKSGD_U_ERN_ALL
  0.008     0.008 #001 RC=   0 Prepare=UPDATE PS_PAY_EARNINGS
                              SET SINGLE_CHECK_USE='N' ,
                              PAY_LINE_STATUS='U' WHERE COMPANY=:1
                              AND PAY_END_DT=:2 AND OFF_CYCLE=:3
                              AND SINGLE_CHECK_USE='P'
                              AND PAY_LINE_STATUS <> 'F'
  0.015     0.000 #001 RC=   0 Bind=0001 Type=SQLPBUF Len=0003 Data=G
  0.000     0.000 #001 RC=   0 Bind=0002 Type=0384 Len=0010 Data=2000
  0.000     0.000 #001 RC=   0 Bind=0003 Type=SQLPBUF Len=0001 Data=N
 74.851    74.851 #001 RC=   0 Execute
```

The Explain showed:

```
QRYNO  QBLKNO  PLANO  METHOD  TNAME   ACCESSTYPE MATCHCOLS ACCESSNAME  INDEXONLY
-----+---------+---------+---------+---------+---------+---------+-----
1012     1       1      0  PS_PAY_EARNINGS   I     0    PSDPAY_EARNINGS     N
1012     1       1      0  PS_PAY_LINE       I     1    PS_PAY_LINE         N
-----+---------+---------+---------+---------+---------+---------+-------------+
PSDPAY_EARNINGS  FIRSTKEYCARD  FULLKEYCARD   CLUSTERRATIO
                 -------+---------+---------+-------------
                    73,623      1,040,557         97
```

```
PS_PAY_LINE      FIRSTKEYCARD  FULLKEYCARD   CLUSTERRATIO
                 -------+---------+---------+-------------
                        26      1,036,120        99
In the test detailed above, these two statements combined to take over 152 seconds out
of 200 seconds (76%) of the total execution time of Paycalc for the Paygroup being
tested.
=====================================================================
```

PAYCONFIRM

This was the longest running statement in Payconfirm: 61.487 seconds out of 186 seconds total for the job.

```
SELECT A.EMPLID ,A.COMPANY ,A.PAYGROUP FROM PS_PAY_EARNINGS A
  WHERE A.COMPANY='GBI' AND A.PAY_END_DT = '2000-05-31'
  AND A.OFF_CYCLE='N' AND A.SINGLE_CHECK_USE
   IN ('N','P') AND A.PAY_LINE_STATUS IN ('I', 'P', 'U')
    AND    (A.OK_TO_PAY='Y' OR NOT EXISTS
      (SELECT 'X' FROM PS_PAY_EARNINGS J WHERE J.COMPANY=A.COMPANY
       AND J.PAYGROUP=A.PAYGROUP AND J.PAY_END_DT=A.PAY_END_DT
       AND J.OFF_CYCLE=A.OFF_CYCLE AND J.EMPLID=A.EMPLID
       AND J.BENEFIT_RCD_NBR=A.BENEFIT_RCD_NBR
       AND J.SINGLE_CHECK_USE IN ('N','P') AND J.OK_TO_PAY='Y')
       AND EXISTS (SELECT 'X' FROM PS_PAY_LINE K
       WHERE K.COMPANY=A.COMPANY AND K.PAYGROUP=A.PAYGROUP
       AND K.PAY_END_DT=A.PAY_END_DT
       AND K.OFF_CYCLE=A.OFF_CYCLE AND K.EMPLID=A.EMPLID
       AND K.BENEFIT_RCD_NBR=A.BENEFIT_RCD_NBR
       AND K.SINGLE_CHECK_USE='C' AND K.CONFIRMED='N' ) ) )
       ORDER BY A.COMPANY ASC ,A.PAYGROUP ASC ,A.EMPLID ASC
   0.000     0.000 #007 RC=   0 SSB=0001 TYPE=SQLPBUF LEN=0011
   0.000     0.000 #007 RC=   0 SSB=0002 TYPE=SQLPBUF LEN=0003
   0.000     0.000 #007 RC=   0 SSB=0003 TYPE=SQLPBUF LEN=0003
   0.000     0.000 #007 RC=   0 Bind=0001 Type=SQLPBUF Len=0003 Data=GB
   0.000     0.000 #007 RC=   0 Bind=0002 Type=0384 Len=0010 Data=2000-
   0.000     0.000 #007 RC=   0 Bind=0003 Type=SQLPBUF Len=0001 Data=N
  61.074    61.074 #007 RC=   0 Execute
   0.000     0.000 #007 RC= 100 Fetch
   0.000     0.000 #007 RC=   0 Close Cursor for PSPCKSGD_S_ERN_CHG
   0.000     0.000 #007 RC=   0 Disconnect
   0.000     0.000 #002 RC= 100 Fetch
   0.001     0.000 #002 RC=   0 Commit
   0.000     0.000 #002 RC=   0 Close Cursor for PSPSTRUN_S_CAL
   0.000     0.000 #002 RC=   0 Disconnect
   0.000     0.000      RC=   0 GETSTMT Stmt=PSPSTRUN_S_CAL
```

The Explain showed:

```
--------+---------+---------+---------+---------+---------+---------+---------+---
QNO  QBLKNO  PLANO  METHOD  TNAME       ACCESSTYPE  MATCHCOLS ACCESSNAME INDEXONLY
---------+---------+---------+---------+---------+---------+---------+---------+--
1013   1       1      0  PS_PAY_EARNINGS     I         0    PSDPAY_EARNINGS N
1013   1       2      3                                0                    N
1013   2       1      0  PS_PAY_EARNINGS     I         4    PSAPAY_EARNINGS Y
1013   3       1      0  PS_PAY_LINE         I         4    PS_PAY_LINE     N
=====================================================================================
```

This was the second longest running statement in Payconfirm: 59.793 seconds out of 144 total.

```
SELECT A.EMPLID ,A.PAYGROUP ,A.PAGE_NUM ,A.LINE_NUM
  FROM PS_PAY_EARNINGS A WHERE A.COMPANY='GBI'
```

```
AND A.PAY_END_DT='2000-05-31' AND A.OFF_CYCLE='N'
AND A.SINGLE_CHECK_USE='N' AND A.PAY_LINE_STATUS='C'
AND A.OK_TO_PAY='Y' AND EXISTS (SELECT 'X'
FROM PS_PAY_EARNINGS B WHERE B.EMPLID=A.EMPLID
AND B.COMPANY=A.COMPANY AND B.PAY_END_DT=A.PAY_END_DT
AND B.OFF_CYCLE=A.OFF_CYCLE AND B.SINGLE_CHECK_USE='C')
ORDER BY A.EMPLID,A.PAYGROUP,A.PAGE_NUM,A.LINE_NUM;
0.000     0.000 #007 RC=   0 SSB=0001 TYPE=SQLPBUF LEN=0011
0.000     0.000 #007 RC=   0 SSB=0002 TYPE=SQLPBUF LEN=0003
0.000     0.000 #007 RC=   0 SSB=0003 TYPE=SQLPSSH LEN=0002
0.000     0.000 #007 RC=   0 SSB=0004 TYPE=SQLPSSH LEN=0002
0.000     0.000 #007 RC=   0 Bind=0001 Type=SQLPBUF Len=0003 Data=GB
0.000     0.000 #007 RC=   0 Bind=0002 Type=0384 Len=0010 Data=2000-
0.000     0.000 #007 RC=   0 Bind=0003 Type=SQLPBUF Len=0001 Data=N
59.793   59.793 #007 RC=   0 Execute
0.000     0.000 #007 RC= 100 Fetch
0.000     0.000 #007 RC=   0 Close Cursor for PSPCKSGD_S_PAY_ERN
0.000     0.000 #007 RC=   0 Disconnect
```

The Explain showed:

```
-----+---------+---------+---------+---------+---------+---------+---------+---------
QRYNO  QBLKNO  PLANNO  METHOD  TNAME       ACCESSTYPE  MATCHCOLS ACCESSNAME INDEXONLY
-----+---------+---------+---------+---------+---------+---------+---------+---------
1014      1       1       0  PS_PAY_EARNINGS     I         0     PSDPAY_EARNINGS  N
1014      1       2       1  PS_PAY_EARNINGS     I         2     PSDPAY_EARNINGS  N
1014      1       3       3                                0                      N
=========================================================================================
```

In the test detailed above, these two statements combined to take over 120 seconds out of 144 seconds (83 percent) of the total execution time of payconfirm for the paygroup being tested.

All four of these worst-performing statements used the PSDPAY_EARNINGS index.

Existing index definitions for PS_PAY_EARNINGS:

```
CREATE UNIQUE INDEX H880TAL.PS_PAY_EARNINGS ON H880TAL.PS_PAY_EARNINGS
(COMPANY, PAYGROUP, PAY_END_DT, OFF_CYCLE, PAGE_NUM, LINE_NUM, ADDL_NBR)
USING STOGROUP PSSHARK PRIQTY 720 SECQTY 720 CLUSTER
   BUFFERPOOL BP4 CLOSE NO;
CREATE  INDEX H880TAL.PSAPAY_EARNINGS ON H880TAL.PS_PAY_EARNINGS
 (COMPANY, PAYGROUP, PAY_END_DT, OFF_CYCLE, PAGE_NUM, LINE_NUM,
  SEPCHK, ADDL_NBR, PAY_LINE_STATUS, OK_TO_PAY, SINGLE_CHECK_USE,
EMPLID, EMPL_RCD, BENEFIT_RCD_NBR)
USING STOGROUP PSSHARK PRIQTY 720 SECQTY 720
   BUFFERPOOL BP4 CLOSE NO;
CREATE  INDEX H880TAL.PSBPAY_EARNINGS ON H880TAL.PS_PAY_EARNINGS
(EMPLID, COMPANY, PAYGROUP, PAY_END_DT, PAY_LINE_STATUS)
 USING STOGROUP PSSHARK PRIQTY 720 SECQTY 720
 BUFFERPOOL BP4 CLOSE NO
CREATE  INDEX H880TAL.PSDPAY_EARNINGS ON H880TAL.PS_PAY_EARNINGS
 (EMPLID,PAY_END_DT)
USING STOGROUP PSSHARK PRIQTY 720 SECQTY 720 CLUSTER
 BUFFERPOOL BP4 CLOSE NO
===========================================================================
```

To reduce the execution times in all of these poorly performing statements a new index, PSCPAY_EARNINGS, was added and isolated into its own bufferpool:

```
CREATE  INDEX H880TAL.PSCPAY_EARNINGS ON H880TAL.PS_PAY_EARNINGS
 (COMPANY, PAY_END_DT, OFF_CYCLE)
```

```
    USING STOGROUP PSSHARK PRIQTY 48000 SECQTY 19200
    BUFFERPOOL BP5 CLOSE NO ;
```

The effect on the worst-performing statement in PAYCALC after this index added:

```
    0.001     0.000 #001 RC=   0 Row Count=000000000
    0.000     0.000 #002 RC= 100 Fetch
    0.000     0.000 #002 RC=   0 Close Cursor for PSPSTRUN_S_CAL
    0.000     0.000 #002 RC=   0 Bind=0001 Type=SQLPBUF Len=0009 Data=KU
    0.000     0.000 #002 RC=   0 Execute
    0.000     0.000 #002 RC=   0 Fetch
    0.000     0.000      RC=   0 GETSTMT Stmt=PSPSTRUN_U_STA1
    0.008     0.008 #001 RC=   0 Prepare=UPDATE PS_PAY_EARNINGS SET PAY_
DT=:3 AND OFF_CYCLE=:4 AND SINGLE_CHECK_USE IN ('N','P') AND PAY_LINE_ST
    0.000     0.000 #001 RC=   0 Bind=0001 Type=SQLPBUF Len=0003 Data=GB
    0.000     0.000 #001 RC=   0 Bind=0002 Type=SQLPBUF Len=0003 Data=KU
    0.000     0.000 #001 RC=   0 Bind=0003 Type=0384 Len=0010 Data=2000-
    0.000     0.000 #001 RC=   0 Bind=0004 Type=SQLPBUF Len=0001 Data=N
    0.001     0.001 #001 RC= 100 Execute  (Was 77.487 seconds, now .001 second)
    0.001     0.000 #001 RC=   0 Row Count=000000000
    0.000     0.000 #001 RC=   0 Commit
    0.000     0.000      RC=   0 GETSTMT Stmt=PSPSTRUN_D_CHK
    0.022     0.022 #001 RC=   0 Prepare=DELETE FROM PS_PAY_CHECK WHERE
 AND NOT EXISTS (SELECT 'X' FROM PS_PAY_EARNINGS B WHERE B.COMPANY=PS_PA
Y_END_DT=PS_PAY_CHECK.PAY_END_DT AND B.OFF_CYCLE=PS_PAY_CHECK.OFF_CYCLE
```

The second slowest statement example was not captured from this run, but similar savings were shown.

The total PAYCALC runtime reduced to .61 minutes from the previous 3.3 minutes.

EXPLAIN confirms new index being used:

```
---------+---------+---------+---------+---------+---------+---------+---------+
QRYNO  QBLK  PLANO  METHOD  TNAME       ACCESSTYPE MATCHCOLS ACCESSNAME INDEXONLY
---------+---------+---------+---------+---------+---------+---------+---------+
1011     1      1      0  PS_PAY_EARNINGS    I         3     PSCPAY_EARNINGS N
PSCPAY_EARNINGS  FIRSTKEYCARD  FULLKEYCARD   CLUSTERRATIO
                 -------+---------+---------+-------------
                      26         1,403          99
=========================================================================================
```

PAYCONFIRM

The effect on the worst-performing statement in PAYCONFIRM after this index was added:

```
    0.015     0.015 #007 RC=   0 COM=SELECT A.EMPLID ,A.COMPANY ,A.PAYGR
T=:2 AND A.OFF_CYCLE=:3 AND A.SINGLE_CHECK_USE IN ('N','P') AND A.PAY_LI
NOT EXISTS (SELECT 'X' FROM PS_PAY_EARNINGS J WHERE J.COMPANY=A.COMPANY
.OFF_CYCLE=A.OFF_CYCLE AND J.EMPLID=A.EMPLID AND J.BENEFIT_RCD_NBR=A.BEN
PAY='Y' ) AND EXISTS (SELECT 'X' FROM PS_PAY_LINE K WHERE K.COMPANY=A.CO
 AND K.OFF_CYCLE=A.OFF_CYCLE AND K.EMPLID=A.EMPLID AND K.BENEFIT_RCD_NBR
='N' ) ) ) ORDER BY A.COMPANY ASC ,A.PAYGROUP ASC ,A.EMPLID ASC
    0.000     0.000 #007 RC=   0 SSB=0001 TYPE=SQLPBUF LEN=0011
    0.000     0.000 #007 RC=   0 SSB=0002 TYPE=SQLPBUF LEN=0003
    0.000     0.000 #007 RC=   0 SSB=0003 TYPE=SQLPBUF LEN=0003
    0.000     0.000 #007 RC=   0 Bind=0001 Type=SQLPBUF Len=0003 Data=GB
    0.000     0.000 #007 RC=   0 Bind=0002 Type=0384 Len=0010 Data=2000-
    0.000     0.000 #007 RC=   0 Bind=0003 Type=SQLPBUF Len=0001 Data=N
    0.002     0.002 #007 RC=   0 Execute (WAS 61.074 seconds, now .002 seconds)
    0.000     0.000 #007 RC= 100 Fetch
    0.000     0.000 #007 RC=   0 Close Cursor for PSPCKSGD_S_ERN_CHG
    0.000      0.000 #007 RC=   0 Disconnect
    0.000      0.000 #002 RC= 100 Fetch
    0.000      0.000 #002 RC=   0 Commit
    0.000      0.000 #002 RC=   0 Close Cursor for PSPSTRUN_S_CAL
```

EXPLAIN confirms new index being used:

```
---------+---------+---------+---------+---------+---------+---------+---------+
QRYNO  QBLK  PLANO  METHOD  TNAME       ACCESSTYPE MATCHCOLS ACCESSNAME INDEXONLY
---------+---------+---------+---------+---------+---------+---------+---------+
1013     1      1      0  PS_PAY_EARNINGS   I          3    PSCPAY_EARNINGS  N
1013     1      2      3                               0                     N
1013     2      1      0  PS_PAY_EARNINGS   I          4    PSBPAY_EARNINGS  N
1013     3      1      0  PS_PAY_LINE       I          4    PS_PAY_LINE      N
---------------------------------------------------------------------------------
```

The effect on the second worst-performing statement in PAYCALC after this index added:

```
    0.042      0.042 #007 RC=   0 COM=SELECT A.EMPLID ,A.PAYGROUP ,A.PAGE
D A.PAY_END_DT=:2 AND A.OFF_CYCLE=:3 AND A.SINGLE_CHECK_USE='N' AND A.PA
FROM PS_PAY_EARNINGS B WHERE B.EMPLID=A.EMPLID AND B.COMPANY=A.COMPANY A
B.SINGLE_CHECK_USE='C') ORDER BY A.EMPLID,A.PAYGROUP,A.PAGE_NUM,A.LINE_N
    0.000      0.000 #007 RC=   0 SSB=0001 TYPE=SQLPBUF LEN=0011
    0.000      0.000 #007 RC=   0 SSB=0002 TYPE=SQLPBUF LEN=0003
    0.000      0.000 #007 RC=   0 SSB=0003 TYPE=SQLPSSH LEN=0002
    0.000      0.000 #007 RC=   0 SSB=0004 TYPE=SQLPSSH LEN=0002
    0.000      0.000 #007 RC=   0 Bind=0001 Type=SQLPBUF Len=0003 Data=GB
    0.000      0.000 #007 RC=   0 Bind=0002 Type=0384 Len=0010 Data=2000-
    0.000      0.000 #007 RC=   0 Bind=0003 Type=SQLPBUF Len=0001 Data=N
    0.007      0.007 #007 RC=   0 Execute    (WAS 59.79)
    0.000      0.000 #007 RC= 100 Fetch
    0.000      0.000 #007 RC=   0 Close Cursor for PSPCKSGD_S_PAY_ERN
    0.000      0.000 #007 RC=   0 Disconnect
    0.000      0.000      RC=   0 GETSTMT Stmt=PSPCKSGD_S_ERN_CHG
    0.015      0.015 #007 RC=   0 COM=SELECT A.EMPLID ,A.COMPANY ,A.PAYGR
```

EXPLAIN confirms new index being used:

```
QRYNO  QBLKNO  PLANNO  METHOD  TNAME       ACCESSTYPE  MATCHCOLS ACCESSNAME  INDEXONLY
-----+---------+---------+---------+---------+---------+---------+---------+---------
1014      1       1       0  PS_PAY_EARNINGS     I         3   PSCPAY_EARNINGS      N
1014      1       2       2  PS_PAY_EARNINGS     I         3   PSCPAY_EARNINGS      N
1014      1       3       3                                0                        N
-------------------------------------------------------------------------------------
```

The total PAYCONFIRM runtime was reduced to .43 minutes, from the previous 2.37 minutes.

Another problem encountered showed severe locking taking place on the following statement when 16 Paycalc processes were run together:

```
SELECT DISTINCT B.PLAN_TYPE,B.BENEFIT_PLAN,B.DEDCD,B.DED_CLASS
  FROM PS_LIMIT_INCLD_TBL L, PS_DEDUCTION_BAL B
   WHERE
      L.LIMIT_TYPE=? AND L.EFFDT=? AND B.EMPLID=? AND B.BALANCE_ID=?
      AND B.BALANCE_YEAR=? AND ((B.PLAN_TYPE<>'00' AND
      B.PLAN_TYPE=L.PLAN_TYPE AND B.BENEFIT_PLAN=L.BENEFIT_PLAN) OR
     (B.PLAN_TYPE='00' AND B.DEDCD=L.DEDCD)) AND
      B.DED_CLASS=L.DED_CLASS
  ORDER BY B.PLAN_TYPE, B.BENEFIT_PLAN, B.DEDCD ,B.DED_CLASS
```

The Explain showed:

```
TNAME              ACCESSTYPE  MATCHCOLS  ACCESSNAME         INDEXONLY
+---------+---------+---------+---------+---------+---------+---------+---------+
PS_DEDUCTION_BAL    I                   1     PS_DEDUCTION_BAL    Y
PS_LIMIT_INCLD_TBL  I                   2     PS_LIMIT_INCLD_TBL  Y
                                        0                         N
                  FIRSTKEYCARDF  FULLKEYCARDF  CLUSTERRATIO
---------+---------+---------+---------+---------+--------
PS_DEDUCTION_BAL       72203       10188419        100
PS_LIMIT_INCLD_TBL         7            940         99
PS_LIMIT_INCLD_TBL  H880TAL   T     H880TALF  BNAPP   940       rows locksize any
PS_DEDUCTION_BAL    H880TAL   T     H880TALP  PYLARG1 10,188,419 ROWS LOCKSIZE ROW
```

Existing index definitions:

```
  UNIQUE INDEX H880TAL.PS_LIMIT_INCLD_TBL
 (LIMIT_TYPE,EFFDT DESC,PLAN_TYPE,BENEFIT_PLAN,DEDCD,DED_CLASS)
  UNIQUE INDEX H880TAL.PS_DEDUCTION_BAL
 (EMPLID,COMPANY,BALANCE_ID,BALANCE_YEAR DESC,BALANCE_QTR DESC,
   BALANCE_PERIOD DESC, BENEFIT_RCD_NBR, PLAN_TYPE, BENEFIT_PLAN,
   DEDCD,DED_CLASS)
```

The primary index on PS_DEDUCTION_BAL was changed to move column COMPANY to be positioned after column BENEFIT_PLAN in an effort to improve MATCHCOLS on this SELECT DISTINCT.

Also added was the second index to hopefully remove sort (order by) on same select distinct.

```
CREATE UNIQUE INDEX H880TAL.PS_DEDUCTION_BAL ON
 H880TAL.PS_DEDUCTION_BAL
  (EMPLID,
   BALANCE_ID,
   BALANCE_YEAR DESC,
   BALANCE_QTR DESC,
   BALANCE_PERIOD DESC,
   BENEFIT_RCD_NBR,
   PLAN_TYPE,
   BENEFIT_PLAN,
   COMPANY,
   DEDCD,
   DED_CLASS) USING STOGROUP PSSHARK
   PRIQTY 240000 SECQTY 19200 CLUSTER
   BUFFERPOOL BP4 CLOSE NO
            ;
CREATE INDEX H880TAL.PS1DEDUCTION_BAL ON
 H880TAL.PS_DEDUCTION_BAL
  (PLAN_TYPE,
   BENEFIT_PLAN,
   DEDCD,
   DED_CLASS) USING STOGROUP PSSHARK
   PRIQTY 240000 SECQTY 19200
   BUFFERPOOL BP4 CLOSE NO
            ;
```

However, a subsequent Explain after this change was made showed that the new index will not be selected by the Optimizer, and therefore the sorting of data could not be reduced by creating this index.

The new Explain does show that MATCHCOLS on the primary index went from 1 to 3.

Next, PLAN_TYPE was moved up ahead of BALANCE_YEAR in an effort to get MATCHCOLS from 3 to 4.

```
WHERE L.LIMIT_TYPE=? AND L.EFFDT=? AND B.EMPLID=? AND B.BALANCE_ID=?
AND B.BALANCE_YEAR=? AND ((B.PLAN_TYPE<>'00' AND
```

However, a new EXPLAIN still showed only 3 columns, not 4 after the change, and the new PS1 index still did not replace the order by sort. Why the DB2 Optimizer chooses not to use these changes is a matter for conjecture.

The first change on the primary index did result in reduced locking and quicker run times, so the change was left in place. The new PS1 index was dropped since it would not be chosen for use by the Optimizer.

RUNSTATS

It is very important from a performance standpoint in a DB2 environment for accurate runstats to exist, and initial runs showed that to certainly be true for this application as well. Initial runstats jobs were run after the database was populated with the 'large model' data from the toolkit. However, each of the three payroll processes (Paysheet, Paycalc and Payconfirm) modify and insert data into a number of tables, and so require updated statistics. Dynamic runstats were turned off for purposes of this benchmark.

Assuming the tablespaces and tables were created with the changes described above, catalog statistics should be updated along the following lines.

After Paysheet has successfully processed, the following statements should be submitted:

```
RUNSTATS TABLESPACE DBNAME.PYAPP8       TABLE INDEX SHRLEVEL REFERENCE
RUNSTATS TABLESPACE DBNAME.PYAPP10     TABLE INDEX SHRLEVEL REFERENCE
RUNSTATS TABLESPACE DBNAME.PYAPP18     TABLE INDEX SHRLEVEL REFERENCE
RUNSTATS TABLESPACE DBNAME.PYLARG5    TABLE INDEX SHRLEVEL REFERENCE
RUNSTATS TABLESPACE DBNAME.PYLARG6    TABLE INDEX SHRLEVEL REFERENCE
RUNSTATS TABLESPACE DBNAME.PYLARG15  TABLE INDEX SHRLEVEL REFERENCE
RUNSTATS TABLESPACE DBNAME.PYLARG24  TABLE INDEX SHRLEVEL REFERENCE
```

After Paycalc has successfully processed, the following statements should be submitted:

```
RUNSTATS TABLESPACE DBNAME.PYAPP3      TABLE INDEX SHRLEVEL REFERENCE
RUNSTATS TABLESPACE DBNAME.PYAPP4      TABLE INDEX SHRLEVEL REFERENCE
RUNSTATS TABLESPACE DBNAME.PYAPP8      TABLE INDEX SHRLEVEL REFERENCE
RUNSTATS TABLESPACE DBNAME.PYAPP18    TABLE INDEX SHRLEVEL REFERENCE
RUNSTATS TABLESPACE DBNAME.PYLARG3    TABLE INDEX SHRLEVEL REFERENCE
RUNSTATS TABLESPACE DBNAME.PYLARG4    TABLE INDEX SHRLEVEL REFERENCE
RUNSTATS TABLESPACE DBNAME.PYLARG7    TABLE INDEX SHRLEVEL REFERENCE
RUNSTATS TABLESPACE DBNAME.PYLARG8    TABLE INDEX SHRLEVEL REFERENCE
RUNSTATS TABLESPACE DBNAME.PYLARG24  TABLE INDEX SHRLEVEL REFERENCE
```

Doing this will significantly improve the performance of initial runs of the subsequent processing steps. Once several 'shakedown' runs were completed and the environment was stabilized, another runstats job was submitted. This job was run after completion of a payroll cycle, when all tables were fully populated relative to the employee base of 90,000, and updated catalog statistics on all tablespaces that were found to contain more than 1,000 rows. Statistics should be monitored on a regular basis for accuracy.

Cluster ratios

Another task to improve performance after 'shakedown' runs was completed and the environment was stabilized to interrogate cluster ratios on the indexes.

A query similar to the following will pull the cluster ratio information for all payroll indexes and display them grouped within the owning tablespace for a subsequent reorg job:

```
   SELECT A.NAME, B.DBNAME, A.TBNAME, B.TSNAME, A.CLUSTERRATIO
                  FROM SYSIBM.SYSINDEXES A, SYSIBM.SYSTABLES B
                      WHERE A.CREATOR = 'DBNAME'
                            AND A.CREATOR = B.CREATOR
                             AND B.NAME = A.TBNAME
                             AND (A.CLUSTERRATIO > 0
                              AND A.CLUSTERRATIO < 95)
            ORDER BY B.TSNAME, A.CLUSTERRATIO;
```

Typically, what will be found after starting with initial data population and then processing a payroll cycle is that a lot of data has been modified or inserted and the indexes are now poorly organized along data lines.

Primary indexes (those beginning with PS_) should ideally have a cluster ratio of 95 or higher. Any that are found to have less, particularly indexes against large primary processing tables like PS_DEDUCTION_BAL or PS_PAY_EARNINGS, must be reorganized to realign the data with the indexes. A reorg job against the corresponding tablespace named in the query should be run, followed by a runstats job to update the system catalog entries.

Monitoring of successive benchmark runs also showed that LEAFDIST can increase to large values on some of the PS_PAY tables. LEAFDIST is the average distance (multiplied by 100) between successive leaf pages during sequential access of an index. If LEAFDIST increases over time, the index should be reorganized. The following queries can be run to monitor candidates for reorganization:

```
          SELECT IXCREATOR, IXNAME, LEAFDIST, INT(CARDF)
               FROM SYSIBM.SYSINDEXPART
                    WHERE IXCREATOR = 'H880xxx'
                          AND LEAFDIST > 50
                    ORDER BY LEAFDIST DESC, IXCREATOR, IXNAME WITH UR;
          SELECT DBNAME, TSNAME
               FROM SYSIBM.SYSTABLEPART
                    WHERE IXCREATOR = 'H880xxx'
                          AND ((CARD > 0 AND (NEARINDREF + FARINDREF) * 100 / CARD > 10)
                             OR PERCDROP  > 10) WITH UR;
          SELECT IXNAME, IXCREATOR, LEAFDIST
               FROM SYSIBM.SYSINDEXPART
                    WHERE LEAFDIST > 50
                          AND IXCREATOR = 'H880xxx'
                     ORDER BY LEAFDIST DESC, IXNAME WITH UR;
```

The query in Example 1 determines whether the rows of a table are stored in the same order as the entries of its clustering index. A large value of FAROFFPOS indicated that the clustering is degenerating. A large value of NEAROFFPOS might also indicate that the tablespace needs reorganizing, but the value of FAROFFPOS is a better indicator.

Example 1   Query

```
SELECT  B.DBNAME, B.TSNAME, A.IXCREATOR, A.IXNAME,
              A.NEAROFFPOS, A.FAROFFPOS
FROM SYSIBM.SYSINDEXPART  A ,
         SYSIBM.SYSTABLES        B,
                       SYSIBM.SYSINDEXES       C
                WHERE IXCREATOR = 'H880xxx'
                      AND (NEAROFFPOS > 10
                         OR FAROFFPOS > 10 )
                      AND C.CLUSTERING = 'Y'
                      AND A.IXNAME =  C.NAME
```

```
                     AND A.IXCREATOR = C.CREATOR
                     AND C.TBNAME = B.NAME
                     AND B.DBNAME LIKE 'PS%'
                ORDER BY DBNAME, TSNAME WITH UR;
SELECT INTABLE.DBNAME, INTABLE.TSNAME
     FROM
         (SELECT B.DBNAME, B.TSNAME
FROM SYSIBM.SYSINDEXPART A ,
            SYSIBM.SYSTABLES  B,
                           SYSIBM.SYSINDEXES C
                   WHERE IXCREATOR = 'H880xxx'
                          AND (NEAROFFPOS > 10
                             OR FAROFFPOS > 10 )
                           AND C.CLUSTERING = 'Y'
                           AND A.IXNAME =  C.NAME
                           AND A.IXCREATOR = C.CREATOR
                           AND C.TBNAME = B.NAME
                           AND B.DBNAME LIKE 'PS%') AS INTABLE
                   GROUP BY INTABLE.DBNAME, INTABLE.TSNAME
      ORDER BY INTABLE.DBNAME ,INTABLE.TSNAME;
```

Miscellaneous settings and suggestions

Some are:

- The default commit frequency was reduced to 25 to help lock concurrency. This value is stored in the column COMMIT_AFTER, which is contained in the PS_INSTALLATION table. This setting affects no other HR applications other than payroll, and works in the reverse of what might be expected—decreasing the number to 25 from the installation default of 300 will cause more frequent checkpoints to be taken, not fewer.
- A situation was encountered where PSSERVERSTAT was causing 913 deadlocks from Process Scheduler during multiple parallel processes. These errors were not returned though the application and could only be observed through a debugging tool like Apptune or by searching the DB2 master logs in SDSF output for the subsystem. If found to be occurring, separating the PSSERVERSTAT table into its own tablespace and setting locksize to ROW should eliminate the problem.
- Indexes for the PS_DEDUCTION_BAL, PS_PAY_EARNINGS, PS_PAY_CHECK and PS_PAY_OTH_EARNS were found to benefit from an increase in free space in previous benchmarks. With 8.8, these and many other objects are now delivered with DDL that has PCTFREE increased to 20 from 10.
- The VSAM datasets for the indexes and tablespaces should be analyzed again after the system has stabilized and tables are fully populated. This is easily done, via TSO 3.4, by entering the high level qualifier for the DSNDBD file names (a list of over 14,000 datasets will be returned for this release of Payroll and HR) and then entering SORT XT™ on the command line. The fourth node in the VSAM dataset name represents the tablespace or index. Tablespaces are easily identified because this node will be the tablespace name, that is, PYAPP7. For indexes, the value in the fourth node translates to the INDEXSPACE column in SYSIBM.SYSINDEXES. Querying INDEXSPACE with this value will return the actual index name to be investigated.
- If the sizing guidelines in this document were followed, and a similar amount of data is loaded (90,000 employees), there should be no values in the XT column greater than 30. In fact, there will only be a handful of files out of the 11,000 that show more than 10 extents taken. Any files found to have taken more than 30 extents, at minimum any of the files representing large tables and indexes like PS_DEDUCTION_BAL, should probably be reallocated for more efficient disk processing.

- If operating in a mixed DASD environment, ensure that PS_DEDUCTION_BAL is on the best disk and channel possible.

Appendix A. Tablespace sizing

TABLESPACE	PRIMARY QUANTITY	SECONDARY QUANTITY
PSIMAGE	PRIQTY 4800	SECQTY 480
PTAMSG	PRIQTY 7200	SECQTY 720
PTAPP	PRIQTY 36000	SECQTY 720
PTAPPE	PRIQTY 48000	SECQTY 720
PTCMSTAR	PRIQTY 1440	SECQTY 14400
PTLOCK	PRIQTY 720	SECQTY 720
PTPRJWK	PRIQTY 14400	SECQTY 1440
PTRPTS	PRIQTY 720	SECQTY 720
PTTBL	PRIQTY 480000	SECQTY 48000
PTTLRG	PRIQTY 960000	SECQTY 96000
PTTREE	PRIQTY 480000	SECQTY 4800
PTWORK	PRIQTY 48000	SECQTY 4800
PTPRC	PRIQTY 1440	SECQTY 720
BNAPP	PRIQTY 921600	SECQTY 48000
BNAPP1	PRIQTY 480000	SECQTY 48000
BNAPP2	PRIQTY 480000	SECQTY 48000
BNAPP3	PRIQTY 480000	SECQTY 48000
BNAPP4	PRIQTY 480000	SECQTY 48000
BNAPP5	PRIQTY 480000	SECQTY 48000
BNAPP6	PRIQTY 480000	SECQTY 48000
BNAPP7	PRIQTY 480000	SECQTY 48000
BNAPP8	PRIQTY 480000	SECQTY 48000
BNLARGE	PRIQTY 921600	SECQTY 48000
BNLARG1	PRIQTY 921600	SECQTY 48000
BNLARG2	PRIQTY 921600	SECQTY 48000
BNLARG3	PRIQTY 921600	SECQTY 48000
BNLARG4	PRIQTY 921600	SECQTY 48000
BNLARG5	PRIQTY 921600	SECQTY 48000
BNLARG6	PRIQTY 921600	SECQTY 48000
CUAUDIT	PRIQTY 480000	SECQTY 48000
CULARGE	PRIQTY 960000	SECQTY 48000
CULARG1	PRIQTY 960000	SECQTY 48000
CULARG2	PRIQTY 960000	SECQTY 48000
CULARG3	PRIQTY 960000	SECQTY 48000
EOAPP	PRIQTY 24000	SECQTY 24000
EOLARGE	PRIQTY 480000	SECQTY 48000
AMAPP	PRIQTY 24000	SECQTY 24000
BDAPP	PRIQTY 24000	SECQTY 24000
FSAPP	PRIQTY 48000	SECQTY 24000
INAPP	PRIQTY 24000	SECQTY 24000
PCAPP	PRIQTY 24000	SECQTY 24000
PCLARGE	PRIQTY 940800	SECQTY 48000
POAPP	PRIQTY 24000	SECQTY 24000
PVAPP	PRIQTY 24000	SECQTY 24000
WAAPP	PRIQTY 24000	SECQTY 24000
FGAPP	PRIQTY 49680	SECQTY 16560
HPAPP	PRIQTY 49680	SECQTY 16560
HRAPP	PRIQTY 828000	SECQTY 48000
HRAPP1	PRIQTY 480000	SECQTY 48000
HRAPP2	PRIQTY 480000	SECQTY 48000
HRAPP3	PRIQTY 480000	SECQTY 48000
HRAPP4	PRIQTY 480000	SECQTY 48000
HRAPP5	PRIQTY 480000	SECQTY 48000
HRAPP6	PRIQTY 480000	SECQTY 48000
HRLARGE	PRIQTY 961968	SECQTY 48000
HRLARG1	PRIQTY 961968	SECQTY 48000
HRLARG2	PRIQTY 961968	SECQTY 48000

```
HRLARG3     PRIQTY 961968    SECQTY 48000
HRLARG4     PRIQTY 961968    SECQTY 48000
HRLARG11    PRIQTY 961968    SECQTY 48000
HTAPP       PRIQTY 108000    SECQTY 24000
PAAPP       PRIQTY 240000    SECQTY 24000
PALARGE     PRIQTY 961968    SECQTY 48000
STAPP       PRIQTY 48000     SECQTY 24000
STLARGE     PRIQTY 961968    SECQTY 48000
GIAPP       PRIQTY 48000     SECQTY 2400
GPAPP       PRIQTY 240000    SECQTY 72000
GPUS001     PRIQTY 24000     SECQTY 7200
GPUS002     PRIQTY 24000     SECQTY 7200
GPUS003     PRIQTY 24000     SECQTY 7200
GPUS004     PRIQTY 24000     SECQTY 7200
GPUS005     PRIQTY 24000     SECQTY 7200
GPUS006     PRIQTY 24000     SECQTY 7200
GPUS007     PRIQTY 24000     SECQTY 7200
GPUS008     PRIQTY 24000     SECQTY 7200
GPUS009     PRIQTY 24000     SECQTY 7200
GPUS010     PRIQTY 24000     SECQTY 7200
GPUS011     PRIQTY 24000     SECQTY 7200
GPUS012     PRIQTY 24000     SECQTY 7200
GPUS013     PRIQTY 24000     SECQTY 7200
GPUS014     PRIQTY 24000     SECQTY 7200
GPUS015     PRIQTY 24000     SECQTY 7200
GPUS016     PRIQTY 24000     SECQTY 7200
GPUS017     PRIQTY 24000     SECQTY 7200
GPUS018     PRIQTY 24000     SECQTY 7200
PIAPP       PRIQTY 48000     SECQTY 7200
PILARGE     PRIQTY 950400    SECQTY 72000
PYAPP       PRIQTY 950400    SECQTY 72000
PYLARGE     PRIQTY 1096224   SECQTY 96000
PYLARG1     PRIQTY 1096224   SECQTY 96000
PYLARG2     PRIQTY 1096224   SECQTY 96000
PYLARG3     PRIQTY 1096224   SECQTY 96000
PYLARG4     PRIQTY 1096224   SECQTY 96000
PYLARG5     PRIQTY 1096224   SECQTY 96000
PYLARG6     PRIQTY 1096224   SECQTY 96000
PYLARG7     PRIQTY 1096224   SECQTY 96000
PYLARG8     PRIQTY 1096224   SECQTY 96000
PYLARG9     PRIQTY 1096224   SECQTY 96000
PYLARG10    PRIQTY 1096224   SECQTY 96000
PYLARG11    PRIQTY 1096224   SECQTY 96000
PYUS001     PRIQTY 240000    SECQTY 72000
PYUS002     PRIQTY 240000    SECQTY 72000
PYUS003     PRIQTY 240000    SECQTY 72000
PYUS004     PRIQTY 240000    SECQTY 72000
TLAPP       PRIQTY 72000     SECQTY 24000
TLLARGE     PRIQTY 828000    SECQTY 72000
TLUS001     PRIQTY 24000     SECQTY 7200
TLUS002     PRIQTY 24000     SECQTY 7200
TLUS003     PRIQTY 24000     SECQTY 7200
TLUS004     PRIQTY 24000     SECQTY 7200
TLUS005     PRIQTY 24000     SECQTY 7200
TLUS006     PRIQTY 24000     SECQTY 7200
HRWORK      PRIQTY 828000    SECQTY 48000
HRWORK1     PRIQTY 480000    SECQTY 24000
HRWORK2     PRIQTY 480000    SECQTY 24000
HRWORK3     PRIQTY 480000    SECQTY 24000
HRWORK4     PRIQTY 480000    SECQTY 24000
```

```
HRWORK5     PRIQTY 480000    SECQTY 24000
HRWORK6     PRIQTY 480000    SECQTY 24000
HRWORK7     PRIQTY 480000    SECQTY 24000
HRWORK8     PRIQTY 480000    SECQTY 24000
HRWORK9     PRIQTY 480000    SECQTY 24000
HRWORK10    PRIQTY 480000    SECQTY 24000
HRWORK11    PRIQTY 480000    SECQTY 24000
HRWORK12    PRIQTY 480000    SECQTY 24000
HRWORK13    PRIQTY 480000    SECQTY 24000
HRWORK14    PRIQTY 480000    SECQTY 24000
HRWORK15    PRIQTY 480000    SECQTY 24000
HRWORK16    PRIQTY 480000    SECQTY 24000
STWORK      PRIQTY 828000    SECQTY 48000
PIWORK      PRIQTY 970704    SECQTY 48000
PYWORK      PRIQTY 970704    SECQTY 48000
TLWORK      PRIQTY 828000    SECQTY 4320
TLWOR001    PRIQTY 48000     SECQTY 24000
TLWOR002    PRIQTY 48000     SECQTY 24000
TLWOR003    PRIQTY 48000     SECQTY 24000
TLWOR004    PRIQTY 48000     SECQTY 24000
TLWOR005    PRIQTY 48000     SECQTY 24000
TLWOR006    PRIQTY 48000     SECQTY 24000
TLWOR007    PRIQTY 48000     SECQTY 24000
TLWOR008    PRIQTY 48000     SECQTY 24000
TLWOR009    PRIQTY 48000     SECQTY 24000
TLWOR010    PRIQTY 48000     SECQTY 24000
TLWOR011    PRIQTY 48000     SECQTY 24000
TLWOR012    PRIQTY 48000     SECQTY 24000
TLWOR013    PRIQTY 48000     SECQTY 24000
TLWOR014    PRIQTY 48000     SECQTY 24000
TLWOR015    PRIQTY 48000     SECQTY 24000
TLWOR016    PRIQTY 48000     SECQTY 24000
TLWOR017    PRIQTY 48000     SECQTY 24000
TLWOR018    PRIQTY 48000     SECQTY 24000
TLWOR019    PRIQTY 48000     SECQTY 24000
TLWOR020    PRIQTY 48000     SECQTY 24000
TLWOR021    PRIQTY 48000     SECQTY 24000
TLWOR022    PRIQTY 48000     SECQTY 24000
TLWOR023    PRIQTY 48000     SECQTY 24000
TLWOR024    PRIQTY 48000     SECQTY 24000
TLWOR025    PRIQTY 48000     SECQTY 24000
TLWOR026    PRIQTY 48000     SECQTY 24000
TLWOR027    PRIQTY 48000     SECQTY 24000
TLWOR028    PRIQTY 48000     SECQTY 24000
TLWOR029    PRIQTY 48000     SECQTY 24000
TLWOR030    PRIQTY 48000     SECQTY 24000
TLWOR031    PRIQTY 48000     SECQTY 24000
TLWOR032    PRIQTY 48000     SECQTY 24000
TLWOR033    PRIQTY 48000     SECQTY 24000
TLWOR034    PRIQTY 48000     SECQTY 24000
TLWOR035    PRIQTY 48000     SECQTY 24000
```

Appendix B. Table and tablespace mappings

```
---------+---------+---------+---------+
  TABLESPACE                  TABLE
---------+---------+---------+---------+
     BNAPP1        PS_BENEFIT_PARTIC
```

```
BNAPP2         PS_LEAVE_ACCRUAL
BNAPP3         PS_LIFE_ADD_BENEFC
BNAPP4         PS_SAVINGS_INVEST
BNAPP5         PS_SAVINGS_BENEFIC
BNAPP6         PS_SAVINGS_PLAN
BNAPP7         PS_DISABILITY_BEN
BNAPP8         PS_VACATION_BEN
BNAPP9         PS_FSA_BENEFIT
BNAPP10        PS_HEALTH_BENEFIT
BNAPP11        PS_HEALTH_DEPENDNT
BNAPP12        PS_LEAVE_PLAN
BNAPP13        PS_LIFE_ADD_BEN
BNAPP24        PS_LIMIT_INCLD_TBL
BNAPP25        PS_LIMIT_IMPIN_TBL
BNAPP26        PS_LIMIT_EXCLD_TBL
BNLARGE        PS_BAS_PARTIC_ELIG
BNLARG1        PS_BAS_PARTIC_COST
BNLARG2        PS_BAS_PARTIC_DPND
BNLARG3        PS_BAS_PARTIC_INVT
BNLARG4        PS_BAS_PARTIC_OPTN
BNLARG5        PS_BAS_PARTIC_PLAN
BNLARG6        PS_BAS_PARTIC
BNLARG7        PS_PRIMARY_JOBS
EOTPAPP        PS_EOTP_BU_TAO
GPAPP          PS_SRC_BANK
HRAPP4         PS_EMPLOYEES
HRAPP7         PS_BOND_LOG
HRAPP8         PS_BOND_SPEC
HRAPP9         PS_BOND_SPEC_DATA
HRAPP10        PS_COMPENSATION
HRAPP12        PS_EMPLMT_WL
HRLARG1        PS_JOB
HRLARG3        PS_PERS_DATA_EFFDT
HRLARG4        PS_EMPLOYMENT
HRLARG5        PS_PERS_NID
HRLARG6        PS_VC_PLAN_MEM
PTPRC1         PSPRCSRQST
PTPRJWK        PSPROJECTWORK
PYAPP1         PS_DED_ARREARS
PYAPP2         PS_GARN_SPEC
PYAPP3         PS_PAY_CAL_BAL_ID
PYAPP4         PS_PAY_CALC_RUNCTL
PYAPP5         PS_PAY_CONF_RUNCTL
PYAPP6         PS_PAY_FORM_TBL
PYAPP7         PS_PAY_GARN_OVRD
PYAPP8         PS_PAY_MESSAGE
PYAPP9         PS_PAYROLL_DATA
PYAPP10        PS_PAYSHEET_RUNCTL
PYAPP11        PS_WRK_CHECK
PYAPP12        PS_WRK_CTX_OVRD
PYAPP13        PS_WRK_DEDUCTION
PYAPP14        PS_WRK_EARNINGS
PYAPP15        PS_WRK_INS_EARNS
PYAPP16        PS_WRK_LINE
PYAPP17        PS_WRK_ONE_TIME
PYAPP18        PS_PAY_CALENDAR
PYAPP20        PS_DIR_DEP_DISTRIB
PYAPP21        PS_DIRECT_DEPOSIT
PYAPP22        PS_GENL_DED_CD
PYLARG1        PS_DEDUCTION_BAL
```

```
PYLARG2       PS_EARNINGS_BAL
PYLARG3       PS_PAY_CHECK
PYLARG4       PS_PAY_DEDUCTION
PYLARG5       PS_PAY_EARNINGS
PYLARG6       PS_PAY_OTH_EARNS
PYLARG7       PS_PAY_SPCL_EARNS
PYLARG8       PS_PAY_TAX
PYLARG9       PS_TAX_BALANCE
PYLARG10      PS_TAX_DIST_EFFDT
PYLARG11      PS_TAX_DISTRIB
PYLARG12      PS_FED_TAX_DATA
PYLARG13      PS_PAY_CTX_OVRD
PYLARG14      PS_PAY_ONE_TIME
PYLARG15      PS_PAY_PAGE
PYLARG16      PS_PAY_TAX_OVRD
PYLARG17      PS_ADDL_PAY_DATA
PYLARG18      PS_ADDL_PAY_EFFDT
PYLARG19      PS_ADDL_PAY_ERNCD
PYLARG20      PS_DEDUCTION_BAL_2
PYLARG21      PS_GENL_DEDUCTION
PYLARG22      PS_PAY_DISTRIBUTN
PYLARG23      PS_PAY_GARNISH
PYLARG24      PS_PAY_LINE
PYLARG25      PS_STATE_TAX_DATA
STWORK        PS_EP_RPT_SNAP_DTL
```

Appendix C. Tablespace lock settings

```
---------+---------+---------+---------+---------+
DBNAME   TABLESPACE      TABLE       LOCKRULE
---------+---------+---------+---------+---------+
H880TALA  PSIMGR    PS_AERUNCONTROL     R
H880TALA  PSIMGR    PS_AERUNCONTROLPC   R
H880TALB  BNAPP8    PS_VACATION_BEN     R
H880TALB  PTAMSG    PSAPMSGMON4VW       R
H880TALB  PTAMSG    PS_AMM_SYNCERR      R
H880TALB  PTAMSG    PS_AMM_SYNCLIST     R
H880TALB  PTAMSG    PS_IB_PUBLIST       R
H880TALB  PTAMSG    PSAPMSGARCHVW       R
H880TALB  PTAMSG    PSAPMSGINUSEVW      R
H880TALB  PTAMSG    PSAPMSGMNARCHVW     R
H880TALB  PTAMSG    PSAPMSGMON1VW       R
H880TALB  PTAMSG    PSAPMSGMON2VW       R
H880TALB  PTAMSG    PSAPMSGDSPSTAT      R
H880TALB  PTAMSG    PSAPMSGPUBATTR      R
H880TALB  PTAMSG    PSAPMSGDOMSTAT      R
H880TALB  PTAMSG    PSAPMSGARCHTMP      R
H880TALB  PTAMSG    PSIBLOGERRP         R
H880TALB  PTAMSG    PSIBLOGHDR          R
H880TALB  PTAMSG    PSIBLOGHDRARCH      R
H880TALB  PTAMSG    PSNODESDOWN         R
H880TALB  PTAMSG    PSSECNODEOPR        R
H880TALB  PTAMSG    PS_AMM_ARCH_PC      R
H880TALB  PTAMSG    PS_AMM_ARCH_PUB     R
H880TALB  PTAMSG    PS_AMM_DETAIL_SRCH  R
H880TALB  PTAMSG    PS_AMM_PUBCONERR    R
H880TALB  PTAMSG    PS_AMM_PUBCONLIST   R
H880TALB  PTAMSG    PS_AMM_PUBERR       R
H880TALB  PTAMSG    PS_AMM_PUBLIST      R
```

```
H880TALB  PTAMSG    PSAPMSGARCHPC       R
H880TALB  PTAMSG    PSAPMSGARCHPH       R
H880TALB  PTAMSG    PSAPMSGXTB          R
H880TALB  PTAMSG    PSIBLOGERR          R
H880TALB  PTAMSG    PSAPMSGPUBINST      R
H880TALB  PTAMSG    PSAPMSGPUBLOCK      R
H880TALB  PTAMSG    PSAPMSGPUBSYNC      R
H880TALB  PTAMSG    PSAPMSGSUBCLOCK     R
H880TALB  PTAMSG    PSAPMSGSUBCSYNC     R
H880TALB  PTAMSG    PSAPMSGPUBCERR      R
H880TALB  PTAMSG    PSAPMSGPUBCERRP     R
H880TALB  PTAMSG    PSAPMSGPUBCLOCK     R
H880TALB  PTAMSG    PSAPMSGPUBCON       R
H880TALB  PTAMSG    PSAPMSGPUBCSYNC     R
H880TALB  PTAMSG    PSAPMSGPUBERR       R
H880TALB  PTAMSG    PSAPMSGPUBERRP      R
H880TALB  PTAMSG    PSAPMSGPUBHDR       R
H880TALB  PTLOCK    PSLOCK              R
H880TALB  PTLOCK    PSVERSION           R
H880TALB  PTRPTS    PS_CDM_NODELANG_VW  R
H880TALB  PTRPTS    PS_CDM_SRCH         R
H880TALB  PTRPTS    PS_MESSAGE_JOB_VW   R
H880TALB  PTRPTS    PS_PA_MSG_JOBID_VW  R
H880TALB  PTRPTS    PS_PA_MSG_PRCS_VW   R
H880TALB  PTRPTS    PS_PMN_AUTHLANG_VW  R
H880TALB  PTRPTS    PS_CDM_FILE_VW_LAN  R
H880TALB  PTRPTS    PS_CDM_FILELIST_VW  R
H880TALB  PTRPTS    PS_CDM_DISTNODE_VW  R
H880TALB  PTRPTS    PS_CDM_AUTH_R_VW    R
H880TALB  PTRPTS    PS_CDM_LIST_VW      R
H880TALB  PTRPTS    PS_CDM_LS_R_VW      R
H880TALB  PTRPTS    PS_PMN_CDM_AUTH_VW  R
H880TALB  PTRPTS    PS_PMN_MSGLOG_VW    R
H880TALB  PTRPTS    PS_CDM_AUTH_U_VW    R
H880TALB  PTRPTS    PS_CDM_LS_NDN_VW    R
H880TALB  PTRPTS    PS_CDM_ARCH_DT_VW   R
H880TALB  PTRPTS    PS_AE_PROCESS_VW    R
H880TALB  PTRPTS    PS_AE_UPGCONV_VW    R
H880TALB  PTRPTS    PS_CDM_TRANSFER     R
H880TALB  PTRPTS    PS_CDM_TRNFR_RJCT   R
H880TALB  PTRPTS    PS_MESSAGE_LOG      R
H880TALB  PTRPTS    PS_CDM_LIST_ARCH    R
H880TALB  PTRPTS    PS_CDM_FILE_EXT     R
H880TALB  PTRPTS    PS_CDM_FILE_LIST    R
H880TALB  PTRPTS    PS_CDM_FILEEXT_LAN  R
H880TALB  PTRPTS    PS_CDM_FILTER       R
H880TALB  PTRPTS    PS_CDM_LIST         R
H880TALB  PTRPTS    PS_CDM_AUTH         R
H880TALB  PTRPTS    PS_CDM_DIST_NODE    R
H880TALC  PTPRC     PS_PRCSOUTTYPE_VW   R
H880TALC  PTPRC     PS_AE_TEMPLOCK_VW   R
H880TALC  PTPRC     PS_AE_TEMPLOCK2_VW  R
H880TALC  PTPRC     PS_OUTDESTFMT_LANG  R
H880TALC  PTPRC     PS_OUTDESTFORMAT    R
H880TALC  PTPRC     PS_OUTDESTSRC       R
H880TALC  PTPRC     PS_OUTDESTSRCLANG   R
H880TALC  PTPRC     PS_OUTDESTTYPE      R
H880TALC  PTPRC     PS_OUTDESTTYPELANG  R
H880TALC  PTPRC     PS_PMN_SRVRLIST     R
H880TALC  PTPRC     PS_CDM_FILTER_ARCH  R
```

```
H880TALC  PTPRC     PS_PRCSDEFNCOND     R
H880TALC  PTPRC     PS_PRCSDEFNMESSAGE  R
H880TALC  PTPRC     PS_AELOCKMGR        R
H880TALC  PTPRC     PS_AETEMPTBLMGR     R
H880TALC  PTPRC     PS_PRCSSEQUENCE     R
H880TALC  PTPRC     PS_PRCSTYPEMETA     R
H880TALC  PTPRC     PS_PRCS_RENSERVER   R
H880TALC  PTPRC     PSPRCSCHLDINFO      R
H880TALC  PTPRC     PS_SERVERNOTIFY     R
H880TALC  PTPRC     PS_SERVERMESSAGE    R
H880TALC  PTPRC     PS_SERVEROPRTN      R
H880TALC  PTPRC     PS_PRCSDEFNCNTDIST  R
H880TALC  PTPRC     PS_PMN_SRVRLSTLANG  R
H880TALC  PTPRC     PS_PRCSDEFNDIST_VW  R
H880TALC  PTPRC     PS_PRCSJOBDIST_VW   R
H880TALC  PTPRC     PS_PRCSJOBITEMDIST  R
H880TALC  PTPRC     PS_PRCSOUTPUT_VW    R
H880TALC  PTPRC     PS_PRCSRQSTITEMDST  R
H880TALC  PTPRC     PSPRCSRQSTARCH      R
H880TALC  PTPRC     PSPRCSRQSTFILE      R
H880TALC  PTPRC     PSPRCSRQSTMETA      R
H880TALC  PTPRC     PSPRCSRQSTTIME      R
H880TALC  PTPRC     PSPRCSRQSTXFER      R
H880TALC  PTPRC     PSPRCSRUNCNTLS      R
H880TALC  PTPRC     PSSERVERSTAT        R
H880TALC  PTPRC     PS_SERVERACTVTY     R
H880TALC  PTPRC     PSPRCSLOCK          R
H880TALC  PTPRC     PSPRCSPARMS         R
H880TALC  PTPRC     PSPRCSQUE           R
H880TALC  PTPRC     PS_PRCSJOBCNTDIST   R
H880TALC  PTPRC     PS_PRCSJOBITEMCHG   R
H880TALC  PTPRC     PS_PRCSJOBMESSAGE   R
H880TALC  PTPRC     PS_PRCSJOBNOTIFY    R
H880TALC  PTPRC     PS_PRCSOUTDESTLIST  R
H880TALC  PTPRC     PS_PRCSOUTDESTTYPE  R
H880TALC  PTPRC     PS_PRCSOUTPUTLIST   R
H880TALC  PTPRC     PS_PRCSRECURDATE    R
H880TALC  PTPRC     PS_PRCSRECUREXEMPT  R
H880TALC  PTPRC     PS_PRCSRQSTDIST     R
H880TALC  PTPRC     PS_PRCSDEFNMETA     R
H880TALC  PTPRC     PS_PRCSDEFNNOTIFY   R
H880TALC  PTPRC     PS_PRCSJOBCHGHIST   R
H880TALC  PTPRC     PS_PRCSRQSTNOTIFY   R
H880TALC  PTPRC     PS_PRCSRUNCNTLDTL   R
H880TALP  PYAPP1    PS_DED_ARREARS      R
H880TALP  PYAPP10   PS_PAYSHEET_RUNCTL  R
H880TALP  PYAPP11   PS_WRK_CHECK        R
H880TALP  PYAPP12   PS_WRK_CTX_OVRD     R
H880TALP  PYAPP13   PS_WRK_DEDUCTION    R
H880TALP  PYAPP14   PS_WRK_EARNINGS     R
H880TALP  PYAPP15   PS_WRK_INS_EARNS    R
H880TALP  PYAPP16   PS_WRK_LINE         R
H880TALP  PYAPP17   PS_WRK_ONE_TIME     R
H880TALP  PYAPP18   PS_PAY_CALENDAR     R
H880TALP  PYAPP2    PS_GARN_SPEC        R
H880TALP  PYAPP3    PS_PAY_CAL_BAL_ID   R
H880TALP  PYAPP4    PS_PAY_CALC_RUNCTL  R
H880TALP  PYAPP5    PS_PAY_CONF_RUNCTL  R
H880TALP  PYAPP6    PS_PAY_FORM_TBL     R
H880TALP  PYAPP7    PS_PAY_GARN_OVRD    R
```

```
H880TALP  PYAPP8    PS_PAY_MESSAGE      R
H880TALP  PYAPP9    PS_PAYROLL_DATA     R
H880TALP  PYLARG1   PS_DEDUCTION_BAL    P
H880TALP  PYLARG12  PS_FED_TAX_DATA     R
H880TALP  PYLARG13  PS_PAY_CTX_OVRD     R
H880TALP  PYLARG14  PS_PAY_ONE_TIME     R
H880TALP  PYLARG15  PS_PAY_PAGE         R
H880TALP  PYLARG16  PS_PAY_TAX_OVRD     R
H880TALP  PYLARG2   PS_EARNINGS_BAL     R
H880TALP  PYLARG24  PS_PAY_LINE         R
H880TALP  PYLARG3   PS_PAY_CHECK        R
H880TALP  PYLARG5   PS_PAY_EARNINGS     R
H880TALP  PYLARG6   PS_PAY_OTH_EARNS    R
H880TALP  PYLARG9   PS_TAX_BALANCE      R
H880TALT  PTPRC1    PSPRCSRQST          R
H880TALT  PTPRC1    PS_PMN_PRCSLIST     R
```

Appendix D. Bufferpool configurations

Table 2 Bufferpool configurations used in benchmark tests

	Z900 process		**Z990 process**
	PeopleSoft Payroll V8.1	PeopleSoft Payroll V8.8	PeopleSoft Payroll V8.8
Buffer name	**Virtual pool**	**Virtual pool**	**Virtual pool**
BP0	2000	10,000	10,000
BP1	5000	10,000	5000
BP2	2000	10,000	2000
BP3	10,000	50,000	30,000
BP4	10,000	50,000	40,000
BP5	5000	20,000	5000
BP6	5000	60,000	60,000
BP7	5000	10,000	10,000
BP8	5000	5000	5000
BP9	5000	30,000	30,000
BP10	10,000	10,000	10,000
BP20	5000	5000	5000
BP21	5000	30,000	30,000
BP22	5000	5000	5000
BP23	5000	30,000	5000
BP24	5000	30,000	5000
Buffer name	**Virtual pool**	**Virtual pool**	**Virtual pool**
BP25	5000	5000	5000

	Z900 process		**Z990 process**
	PeopleSoft Payroll V8.1	PeopleSoft Payroll V8.8	PeopleSoft Payroll V8.8
BP26	5000	10,000	5000
BP27	5000	5000	5000
BP28	5000	5000	5000
BP29	5000	5000	5000
BP31		20,000	120,000
BP32	240	20,000	30,000
BP33		40,000	100,000
BP34		50,000	100,000
BP35		40,000	100,000
BP32K		240	240
BP32K1		1000	10,000
BP32K2		500	
BP32K3		500	
BP32K4		300	
BP16K0		500	

The team that wrote this Redpaper

This Redpaper was produced by specialists working at the Peoplesoft Headquarters in Pleasanton California and the IBM International Technical Support Organization (ITSO) in Poughkeepsie, New York.

John Gray has been a Senior Performance Engineer at PeopleSoft in California for the last two years. He has 28 years of experience in IMS™ and DB2 as a database programmer and performance analyst.

Lydia Parziale is a Staff Software Engineer for the IBM Watson Information Systems Group at the Watson Research Center in Yorktown, New York. She is a Project Leader with domestic and international experience in technology management including software development, project leadership, and strategic planning. Her areas of expertise include e-Business development and database management technologies.

Thanks to the following for their help in this project:

Michael Curtis
IBM Advanced Technical Support Center, Oakland

Mike Ebbers
IBM International Technical Support Organization, Poughkeepsie

Notices

This document created or updated on December 22, 2003.

Send us your comments in one of the following ways:

- Use the online **Contact us** review redbook form found at:
  ibm.com/redbooks
- Send your comments in an Internet note to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYJ Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks