TECHNICAL MANUAL

MANAGING ORACLE TABLE PARTITIONING IN PEOPLESOFT APPLICATIONS WITH GFC_PSPART PACKAGE

Prepared By David Kurtz, Go-Faster Consultancy

Technical Manual

Version 1.10 – http://www.go-faster.co.uk/gfc_pspart.manual.pdf

Friday 8 October 2021

(E-mail: david.kurtz@go-faster.co.uk, telephone +44-7771-760660)

File: gfc_pspart.manual.docx, 8 October 2021

Contents

Introduction 3
Delivered Files 4
Installation 6
Operation 7
  Example Output 9
Meta-Data 12
 Meta-data Tables 12
  GFC_PART_INDEXES 12
  GFC_PART_LISTS 14
  GFC_PART_TABLES 15
  GFC_PART_RANGES 18
  GFC_PART_SUBPARTS 19
  GFC_PART_RANGE_LISTS 19
  GFC_PS_IDXDDLPARM 20

GFC_PS_INDEXDEFN 20
GFC_PS_KEYDEFN 21
GFC_TEMP_TABLES 22
Working Storage Tables 23
GFC_PS_TABLES 23
GFC_PS_TAB_COLUMNS 24
GFC_ORA_TAB_COLUMNS 25
GFC_DDL_SCRIPT 25
Context 26
Example Meta-Data 28
Example General Ledger Meta-Data 28
Example Global Payroll Meta-Data 30
Global Indexes 36
Global Payroll Streams 37
Function Based Indexes 38
List Partitioning 40
Selectively Building Sub-Partitions 42
Setting Context Values 43
Globally Partitioned Indexes 43
Example Time & Labor Meta-Data 44
Archiving Time-Based Range Partitioning 44
Meta-Data Report 46
Partitioned Tables 46
Range Partitioning 46
List Partitioning 47
Global and Globally Partitioned Indexes 47
Specified Tablespaces 48

Package GFC_PSPART 49

BANNER Procedure 49
DISPLAY_DEFAULTS Procedure 50
HISTORY Procedure 51
RESET_DEFAULTS Procedure 52
SET_DEFAULTS Procedure 52
TRUNCATE_TABLES Procedure 54
SPOOLER Function 55
MAIN Procedure 56

Introduction

This document is a guide to the PeopleSoft Partitioning utility package; GFC_PSPART.

Although table partitioning is usually presented as performance option for Data Warehouse systems, it can also be used very successfully in OLTP systems such as PeopleSoft. Even official PeopleSoft documentation such as the Red Paper *'PeopleSoft 8 Batch Performance on Oracle Database'* discusses the merits of partitioning[1]. However, that paper completely fails to mention that PeopleSoft's Application Designer will not build the DDL to build partitioned objects. If you want to partition tables in your application, you are own your own facing the prospect of manually maintaining DDL scripts.

The GFC_PSPART PL/SQL package dynamically generates the DDL to create and manage partitioned and Global Temporary tables and indexes in PeopleSoft systems based on the meta-data in the PeopleTools tables and addition meta-data that describes the partitioning strategy. These generated scripts replace the use of the Application Designer to build DDL scripts.

Partitioning can also be a segue into archiving. Older historical data in separate partitions can be archived by dropping the partition or changing it with another table (and possibly exchanging it again into another partition. GFC_PSPART can suppress generation of partitions that have been archived and generate a partitioned copy of a partitioned table with only the partitions necessary to hold archived data.

If you are partitioning output tables to prevent read-consistency contention between concurrent processes, then temporary working storage tables must also be addressed. However, rather than partition them it is an option to make them Global Temporary tables. The GFC_PSPART package can also generate the DDL for this.

Essentially, GFC_PSPART can be considered to be a replacement for the build script builder in Application Designer for PeopleSoft records that are to be built as Partitioned and Global Temporary tables.

---

[1] However, this document doesn't properly explain the advantages of partitioning. The reduction in I/O can be achieved by:

i. Partition elimination to reduce scans of tables and indexes.

ii. Separation of data to reduce consistent read during parallel batch processing..

Delivered Files

The package is written in PL/SQL. It is composed of a number of SQL scripts. The scripts are designed to run in Oracle's SQL*Plus utility only. It is not supported to use any other tool.

- *gfcbuildpkg.sql*: This script installs the package.
    - builds the package header – the package header has not been wrapped and the parameters to the public procedures and functions in the package are documented here.
    - runs *gfcbuildpkgbody.plb* to create the package
    - Sets default values in a database context
- *gfcbuildtab.sql*: This script contains the DDL to build the working storage tables used by the package.
- *gfcbuildpkgbody.plb*: The GFC_PSPART package has been supplied as a 'wrapped' package. The source code is not available.
- *partdata.sql:* The GFC_PSPART package is almost entirely driven by meta-data. Most of it comes from the PeopleTools tables, but the definition of which tables to partition, how they should be partitioned, how their indexes should be partitioned, and where those partitions should be built is. Partitioning configuration on some systems can become quite complicated. Separate scripts to build meta-data for each functional area have been created (such as *gp-partdata.sql*) and are called from this script. This script also generates a summary report of the partitioning meta-data after it has been loaded (see Appendx Example Meta-Data from page 28).
- *gp-partdata.sql*: This utility was originally developed for use with PeopleSoft Global Payroll Systems were Payroll 'Streaming[2]' is in use. This file builds the meta-data for the partitioning of Global Payroll tables only. The configuration required is almost identical for most customers so a standard configuration is supplied as a starting point. It is expected that this will then be customised as necessary.
- *ppm-partdata.sql*: From PeopleTools 8.48 LONG columns can be replaced with CLOBs. This script builds the meta-data to specify partitioning for the tables used by PeopleSoft Performance Monitor to store performance metrics. This can significantly improve the performance of analytics queries. This script is supplied as an example.
- *gfcbuild.sql*: Is a script to reload the partitioning meta-data, call the GFC_PSPART package and spool the output to file (see Operation on page 7).
- *gfcbuildone.sql*: This is a variant of *gfcbuild.sql* which builds the DDL scripts for named tables only (see Operation on page 7).

[2] PeopleSoft coined this term to describe running certain payroll processes in several concurrently executing processes.

- *gfcbuildspool.sql:* This script spools the output of the GFC_PSPART package from a temporary working storage table to a file. This approach was used so that the package can be run by a developer from a client PC.

- *Psownerid.sql:* This script determines the name of the PeopleSoft Owner ID account (usually SYSADM), so it can be referenced in subsequent scripts.

Installation

The package is designed to be installed and run in the owner of the PeopleSoft database – by default SYSADM[3]. The scripts are designed to be run in SQL*Plus. All the files should be placed in the same directory, and SQL*Plus should be run in that directory.

If running on a windows client, you may experience problems if there is a space in the name of or path to this directory. SQL scripts may not be able to execute other SQL scripts. The workaround is make that directory the working directory in the shortcut to SQL*Plus. It can be helpful to have a dedicated SQL*Plus shortcut in the working directory.

1. This account must be explicitly granted the following privileges rather than via a role by running *gfcbuildpriv.sql*:

   - *SELECT ON sys.v_$version* – some aspects of generated SQL vary with the version of Oracle. For example, from Oracle 10g the DROP TABLE statements can optionally include the PURGE option.

   - *SELECT ON sys.v_$parameter*

   - *CREATE ANY CONTEXT* – the package uses a context rather than a database table to hold settings persistently that modify its behaviour. This privilege is only required temporarily during installation, and can be revoked again afterwards.

   - *SELECT ON sys.dba_tables* – the package checks which table already exist by querying this catalogue view.

   - *SELECT ON sys.dba_tab_partitions* – the package checks which table partitions already exist by querying this catalogue view.

   - *SELECT ON sys.dba_ind_partitions* – the package checks which index partitions already exist by querying this catalogue view.

2. The working storage and meta-data tables are created by running the script *gfcbuildtab.sql.*
3. The package can be installed by running the *gfcbuildpkg.sql* script (which in turn calls and *gfcbuildpkgbody.plb*).

[3] If you have chosen to rename the PeopleSoft database owner user it will be necessary to change any explicit references to this schema in the scripts.

Operation

The delivered scripts *gfcbuild.sql* and *gfcbuildone.sql* demonstrate how the package can be used to rebuild all or most tables.

Essentially this process is a replacement for building DDL scripts in Application Designer. The main procedure builds the DDL and puts it into the working storage table and the spools it out to a number of files. The procedure can be run for all tables, or for some tables. It can be run several times for different tables to build up a single set of scripts which handle a specified list of tables.

```
rem gfcbuild.sql
rem (c) Go-Faster Consultancy Ltd.
clear screen

spool gfcbuild

execute gfc_pspart.truncate_tables(p_all=>TRUE);[4]
@@partdata[5]
--execute gfc_pspart.truncate_tables;[6]

--all tables
execute gfc_pspart.main;[7]

--just generate global temporary tables
--execute gfc_pspart.main(p_rectype => 'T');[8]

--just generate named tables
--execute gfc_pspart.main(p_recname => 'GP_PI_GEN_SOVR', p_rectype => 'P');[9]

--extract script to file
@@gfcbuildspool.sql[10]
set head on feedback on termout on pages 50
```

[4] All the meta-data and working storage tables are being truncated. It is only necessary to do this if the meta-data has changed and therefore needs to be reloaded.

[5] The *partdata.sql* script (see Example Meta-Data on page 27) reloads the meta-data. Other meta-data scripts are called from within this script.

[6] It is only necessary to run truncate_tables on the working storage tables if it has not already been run to clear all tables.

[7] Running the main procedure with no paramters builds the DDL for ALL tables specified in the meta-data.

[8] It is possible to build the DDL just some tables, this commented out command would build

[9] This command generates the DDL for a specified partitioned table.

[10] Finally, the generated DDL is spooled from the working storage table to a local file. This approach is used so that the utility can be used on a client rather than just a database server.

This is an alternative version of gfcbuild.sql that is only going to build the DDL for a number of named tables.

```
rem gfcbuildone.sql
rem (c) Go-Faster Consultancy Ltd.
clear screen

spool gfcbuild

--execute gfc_pspart.truncate_tables(p_all=>TRUE);[11]
--set termout off
--@@partdata
--set termout on
execute gfc_pspart.truncate_tables;[12]

--just generate named tables
--execute gfc_pspart.main(p_recname => 'GP_RSLT_ACUM', p_rectype => 'P');
--execute gfc_pspart.main(p_recname => 'GP_RSLT_PIN', p_rectype => 'P');
execute gfc_pspart.main(p_recname => 'GP_RSLT%', p_rectype => 'P');[13]
execute gfc_pspart.main(p_recname => 'TL_PAYABLE_TIME', p_rectype => 'P');
execute gfc_pspart.main(p_recname => 'TL_RPTD_TIME');
execute gfc_pspart.main(p_projectname => 'GFC_PROJECT');[14]

--pause
--extract script to file
@@gfcbuildspool.sql
```

[11] In this case the meta-data is not being truncated and reloaded, because it has not changed.

[12] Instead, we will only truncate the working storage tables.

[13] In this case all tables whose names begin with GP_RSLT. This is an alternative to explicitly listing them.

[14] In this case, the DDL for all the records that appear in the named project for which there is also partition meta-data will be generated. The package still generates an output project with the name specified in the context (see Context on page 22).

If you run the main procedure for a record that has already been generated since you last cleared the working storage tables, you will generate a unique constraint error.

```
SQL> execute gfc_pspart.main(p_recname => 'GP_PI_GEN_SOVR');

PL/SQL procedure successfully completed.

SQL> execute gfc_pspart.main(p_recname => 'GP_PI_GEN_SOVR');
BEGIN gfc_pspart.main(p_recname => 'GP_PI_GEN_SOVR', p_rectype => 'P'); END;

*
ERROR at line 1:
ORA-00001: unique constraint (SYSADM.GFC_PS_TABLES) violated
ORA-06512: at "SYSADM.GFC_PSPART", line 334
ORA-06512: at "SYSADM.GFC_PSPART", line 3467
ORA-06512: at line 1
```

Example Output

This is an example of the script generate by the package.

```
spool gfcbuild_hcm89_tl_payable_time.lst[15]
REM Generated by GFC_PSPART - (c)Go-Faster Consultancy Ltd. www.go-faster.co.uk 2001-2010
REM HCM89 @ 13:09:58 02.10.2010[16]
…
WHENEVER SQLERROR CONTINUE[17]
ALTER TABLE sysadm.ps_tl_payable_time RENAME PARTITION tl_payable_time_001 TO old_tl_payable_time_001;
…
DROP INDEX ps_tl_payable_time;
…

WHENEVER SQLERROR EXIT FAILURE
CREATE TABLE sysadm.gfc_tl_payable_time
(emplid VARCHAR2(11) NOT NULL
…
)
TABLESPACE GPAPP
PCTUSED 95 PCTFREE 1
PARTITION BY RANGE(EMPLID)
(PARTITION tl_payable_time_001 VALUES LESS THAN ('KUL451') TABLESPACE GPSTRM01TAB
…
,PARTITION tl_payable_time_010 VALUES LESS THAN (MAXVALUE) TABLESPACE GPSTRM10TAB
)
```

[15] A spool file is generated for each table processed within each generated script, thus there is a detailed log of what has been performed by the generated scripts.

[16] This comment shows the name of the PeopleSoft database against which the script was generated, and the date and time at which it was generated.

[17] The scripts are designed to be run in SQL*Plus. They use the WHENEVER SQLERROR command to control whether SQL*Plus terminates after a SQL error in a command. Thus the script is failsafe, and can be left to run unattended. It may error, but it will never destroy data because it continues to execute after an error.

```
ENABLE ROW MOVEMENT
PARALLEL
NOLOGGING
/

GRANT SELECT ON sysadm.gfc_tl_payable_time TO gfc_read_only
/
GRANT INSERT, UPDATE, DELETE ON sysadm.gfc_tl_payable_time TO gfc_update
/

LOCK TABLE sysadm.ps_tl_payable_time IN EXCLUSIVE MODE
/

CREATE OR REPLACE TRIGGER sysadm.tl_payable_time_nochange[18]
BEFORE INSERT OR UPDATE OR DELETE ON sysadm.ps_tl_payable_time
BEGIN
   RAISE_APPLICATION_ERROR(-20100,'NO DML OPERATIONS ALLOWED ON SYSADM.PS_TL_PAYABLE_TIME');
END;
/

ALTER TABLE sysadm.ps_tl_payable_time PARALLEL
/
LOCK TABLE sysadm.ps_tl_payable_time IN EXCLUSIVE MODE[19]
/

BEGIN
INSERT /*+APPEND NOLOGGING*/ INTO sysadm.gfc_tl_payable_time (…
) SELECT …
FROM sysadm.ps_tl_payable_time;
COMMIT;
END;
/
…
CREATE UNIQUE INDEX sysadm.gfc_tl_payable_time ON sysadm.gfc_tl_payable_time
(emplid
,empl_rcd
,dur DESC
,seq_nbr
)
LOCAL
(PARTITION tl_payable_time_001 TABLESPACE GPSTRM01IDX
…
,PARTITION tl_payable_time_010 TABLESPACE GPSTRM10IDX
)
TABLESPACE PSINDEX
PCTFREE 1
PARALLEL
```

[18] A trigger is created on the table being copied to prevent any updates occurring between now and when the new copy replaces it. This will, quite deliberately, cause any attempted updates to fail. It will not prevent direct path inserts, but PeopleSoft doesn't do this.

[19] The source table is explicitly locked in case any transactions were in flight when the trigger was created.

```
NOLOGGING
/

ALTER INDEX sysadm.gfc_tl_payable_time
LOGGING
/
ALTER INDEX sysadm.gfc_tl_payable_time
NOPARALLEL
/

WHENEVER SQLERROR CONTINUE
DROP INDEX sysadm.ps_tl_payable_time
/
WHENEVER SQLERROR EXIT FAILURE
ALTER INDEX sysadm.gfc_tl_payable_time RENAME TO ps_tl_payable_time[20]
/

WHENEVER SQLERROR EXIT FAILURE
ALTER TABLE sysadm.gfc_tl_payable_time LOGGING NOPARALLEL MONITORING
/
WHENEVER SQLERROR CONTINUE
ALTER TABLE sysadm.ps_tl_payable_time RENAME TO old_tl_payable_time
/

WHENEVER SQLERROR EXIT FAILURE
ALTER TABLE sysadm.gfc_tl_payable_time RENAME TO ps_tl_payable_time
/

WHENEVER SQLERROR CONTINUE
DROP TABLE sysadm.old_tl_payable_time PURGE
/

ALTER TRIGGER PSFT_DDL_LOCK ENABLE
/

DROP TRIGGER sysadm.tl_payable_time_nochange
/
```

[20] The newly created objects replace the old ones through a series or rename operations, and the old objects are dropped.

A P P E N D I X

Meta-Data

Not surprisingly, there is nowhere in the PeopleTools tables to hold partitioning information, so additional tables have been created. It is necessary to understand what the meta-data that in tables means, so that you can specify your own meta-data.

Meta-data Tables

GFC_PART_INDEXES

This table can be used to specify any indexes that are not to be locally partitioned. Any indexes for which there is no entry in this table will be locally partitioned.

Column	**Datatype**	**Null / Constraint**	**Description**
RECNAME*	VARCHAR2(15)	NOT NULL	PeopleSoft Record Name
INDEXID*	VARCHAR2(1)	NOT NULL	Identifier of PeopleSoft Index
PART_ID	VARCHAR2(8)	NOT NULL	ID of partitioning strategy. Many tables can share one partitioning strategy.
PART_COLUMN	VARCHAR2(100)	NOT NULL	Range partitioning column, or comma separated columns.
PART_TYPE	VARCHAR2(1)	NOT NULL, R, L, H or N	Partitioning Type: (R)ange or (L)ist (H)ash, or (N)ot partitioned
SUBPART_ID	VARCHAR2(8)	NOT NULL	ID of sub-partitioning strategy. If not specified set to same value as PART_ID
SUBPART_TYPE	VARCHAR2(1)	R, L, H, N	Subpartitioning Type: (L)ist or (H)ash. Default: (N)ot sub-partitioned
SUBPART_COLUMN	VARCHAR2(100)		Name of column on which to sub-partition.
HASH_PARTITIONS	NUMBER	NOT NULL > 0	Number of Hash partitions. Default 0
IDX_TABLESPACE	VARCHAR2(30)		Tablespace to be specified at

			index level.
IDX_STORAGE	VARCHAR2(100)		Storage Clause to be specified at index level.
OVERRIDE_SCHEMA	VARCHAR2(30)		Schema in which table to be built. Used if not building object in schema of PeopleSoft Owner ID.
NAME_SUFFIX	VARCHAR2(20)		Suffix to be added to Index and partition names
PARTIAL_INDEX	VARCHAR2(1)	Y, N	Allow partitial indexing.

GFC_PART_LISTS

This table specifies the list partition definitions. It contains one row for each list partition defined.

Column	**Datatype**	**Null / Constraint**	**Description**
PART_ID*	VARCHAR2(8)	NOT NULL	ID of partitioning strategy. Many tables can share one partitioning strategy.
PART_NO*	NUMBER	NOT NULL	Controls Sequence List Partition specification in DDL
PART_NAME	VARCHAR2(30)	NOT NULL	This value is appended to the RECNAME name to make the partition name
LIST_VALUE	VARCHAR2(100)	NOT NULL	List partition values.
TAB_TABLESPACE	VARCHAR2(30)		Tablespace to be specified at table partition level.
IDX_TABLESPACE	VARCHAR2(30)		Tablespace to be specified at index partition level.
TAB_STORAGE	VARCHAR2(100)		Storage Clause to be specified at table partition level.
IDX_STORAGE	VARCHAR2(100)		Storage Clause to be specified at index partition level.

ARCH_FLAG	VARCHAR2(1)	A, N, D	Enable archiving feature
			
			A=Archive
			
			D=Purge by dropping partitions
			
			N=Do not archive

Additional Unique Constraint: PART_ID, PART_NAME

GFC_PART_TABLES

This table contains one row for each record whose corresponding database table is to be partitioned.

Column	Datatype	Null / Constraint	Description
RECNAME*	VARCHAR2(15)	NOT NULL	PeopleSoft Record Name
PART_ID	VARCHAR2(8)	NOT NULL	ID of partitioning strategy. Many tables can share one partitioning strategy.
PART_COLUMN	VARCHAR2(100)	NOT NULL	Range partitioning column, or comma separated columns.
PART_TYPE	VARCHAR2(1)	NOT NULL, R, L, H, I, N	Partitioning Type: (R)ange, (L)ist or (H)ash, (I)nterval or (N)ot Partitioned
INTERVAL_EXPR	VARCHAR2(100)		Interval Partitioning Expression
SUBPART_TYPE	VARCHAR2(1)	L, H, N	Subpartitioning Type: (L)ist or (H)ash. Default: (N)ot Partitioned
SUBPART_COLUMN	VARCHAR2(100)		Name of column on which to sub-partition.
HASH_PARTITIONS	NUMBER	NOT NULL >= 0	Number of Hash partitions. Default 0
TAB_TABLESPACE	VARCHAR2(30)		Tablespace to be specified at table level.
IDX_TABLESPACE	VARCHAR2(30)		Tablespace to be specified at index level.
TAB_STORAGE	VARCHAR2(100)		Storage Clause to be specified at table level.
IDX_STORAGE	VARCHAR2(100)		Storage Clause to be specified at index level.
STATS_TYPE	VARCHAR2(1)	Y, N, D	Y: Collect statistics D: Delete Statistics N: Do Not Collect Statistics
SAMPLE_SIZE	NUMBER		As sample size in DBMS_STATS, except that NULL indicates that DBMS_STATS.AUTO_SAMPLE_SIZE will be used.

METHOD_OPT	VARCHAR2(100)		Passed to METHOD_OPT parameter of DBMS_STATS procedures
OVERRIDE_SCHEMA	VARCHAR2(30)		Schema in which table to be built. Used if not building object in schema of PeopleSoft Owner ID
CRITERIA	VARCHAR2(1000)		Complete WHERE clause with logical condition that will appended to the INSERT … SELECT statement that copies data from original table into new table[21].
ARCH_FLAG	VARCHAR2(1)	A, N, D	Enable archiving feature A=Archive D=Purge by dropping partitions N=Do not archive Note that partitions will only be archived if both this flag and GFC_PART_RANGES.ARCH_FLAG are set.
ARCH_SCHEMA	VARCHAR2(30)		Schema in which the archive table is to be built. Explicit SELECT privilege will be granted to the PeopleSoft Owner ID
ARCH_RECNAME	VARCHAR2(15)		Record name to be used to determine name of archive table. If null the table name will be determined by ARCH_TABLE_NAME. To be used if the archive table is to be created in the PeopleSoft Owner's schema.
ARCH_TABLE_NAME	VARCHAR2(30)		Name of Archive Table. If null, the archive table will have the same name as the base table.
NOARCH_CONDITION	VARCHAR2(1000)		Logical condition to identify the rows that should be preserved during an archive or purge process by moving them back the table after they have been exchanged out during the archive process.

[21] This parameter can be used when you want to archive data and also reorganise the partitioned tables.

Other Constraints:

- PART_COLUMN and SUBPART_COLUMN may not both be null
- The type of partitioning may not be the same as the sub-partitioning

GFC_PART_RANGES

This table specifies the range partition definitions. It contains one row for each range defined.

Column	Datatype	Null / Constraint	Description
PART_ID*	VARCHAR2(8)	NOT NULL	ID of partitioning strategy. Many tables can share one partitioning strategy.
PART_NO*	NUMBER	NOT NULL	Sequence Number of range partition
PART_NAME	VARCHAR2(30)	NOT NULL	This value is appended to the RECNAME name to make the partition name
PART_VALUE	VARCHAR2(100)	NOT NULL	Range less than value for partition
TAB_TABLESPACE	VARCHAR2(30)		Tablespace to be specified at table partition level.
IDX_TABLESPACE	VARCHAR2(30)		Tablespace to be specified at index partition level.
TAB_STORAGE	VARCHAR2(100)		Storage Clause to be specified at table partition level.
IDX_STORAGE	VARCHAR2(100)		Storage Clause to be specified at index partition level.

ARCH_FLAG	VARCHAR2(1)	A, D, N	N=Do Not Archived (default)
			
			A=Archive
			
			D= Purge by dropping partitions
			
			Note that partitions will only be archived if both this flag and GFC_PART_TABLES.ARCH_FLAG are set.

Additional Unique Constraint: PART_ID, PART_NAME

GFC_PART_SUBPARTS

It is assumed that in a RANGE-LIST partitioned table all combinations should be built. But this may not always be the case. Where a combination is not to be built, a row with a can be added to this table with the value for BUILD set to N.

Column	Datatype	Null / Constraint	Description
PART_ID*	VARCHAR2(8)	NOT NULL	ID of partitioning strategy. Many tables can share one partitioning strategy.
PART_NAME	VARCHAR2(30)	NOT NULL	Name of partition in GFC_PART_RANGES or GFC_PART_LISTS
SUBPART_ID	VARCHAR2(8)	NOT NULL	ID of subpartitioning strategy. If not specified defaults to same value PART_ID.
SUBPART_NAME	VARCHAR2(30)	NOT NULL	Name of subpartition in GFC_PART_RANGES or GFC_PART_LISTS
BUILD	VARCHAR2(1)	NOT NULL	(Y)es - default (N)o.

GFC_PART_RANGE_LISTS

This object is now a view based on GFC_PART_SUBPARTS for backward compatibility.

Column	Datatype	Null / Constraint	Description
PART_ID*	VARCHAR2(8)	NOT NULL	ID of partitioning strategy. Many tables can share one partitioning strategy.
RANGE_NAME	VARCHAR2(30)	NOT NULL	Name of Range partition in GFC_PART_RANGES
LIST_NAME	VARCHAR2(30)	NOT NULL	Name of List partition in GFC_PART_LISTS
BUILD	VARCHAR2(1)	NOT NULL	(Y)es - default (N)o.

GFC_PS_IDXDDLPARM

It is not possible to define function based indexes in PeopleSoft. Sometimes it is also necessary to build such indexes on partitioned tables, and sometimes to partition them. This table is used to specify the DDL override information that would otherwise come from the PeopleTools table PSIDXDDLPARM.

Column	Datatype	Null / Constraint	Description
RECNAME*	VARCHAR2(15)	NOT NULL	PeopleSoft Record Name
INDEXID*	VARCHAR2(1)	NOT NULL	Identifier of PeopleSoft Index
PARMNAME*	VARCHAR2(8)	NOT NULL	Name of DDL Model Parameter
PARMVALUE	VARCHAR2(128)	NOT NULL	Value to be substituted into DDL model.

GFC_PS_INDEXDEFN

A working storage table that contains details of both the indexes defined in the meta-data for this package and in PeopleTools. Meta-data can be inserted into this table directly if it is necessary to define indexes that are not defined in PeopleSoft. For example, function based indexes.

Column	Datatype	Null / Constraint	Description
RECNAME*	VARCHAR2(15)	NOT NULL	PeopleSoft Record Name
INDEXID*	VARCHAR2(1)	NOT NULL	Identifier of PeopleSoft Index
SUBRECNAME	VARCHAR2(15)	NOT NULL	Subrecord from which field definition derived.
SUBINDEXID	VARCHAR2(1)	NOT NULL	Identifier of PeopleSoft Index on subrecord
PLATFORM_ORA	NUMBER	NOT NULL	=1 if index to be build on Oracle database
CUSTKEYORDER	NUMBER	NOT NULL	= 1 if Custom Key Order
UNIQUEFLAG	NUMBER	NOT NULL	= 1 if index to be build unique

GFC_PS_KEYDEFN

It is not possible to define function based indexes in PeopleSoft. Sometimes it is also necessary to build such indexes on partitioned tables, and sometimes to partition them. This table is used to specify the keys on such indexes that would otherwise come from the PeopleTools table PSKEYDEFN.

Column	**Datatype**	**Null / Constraint**	**Description**
RECNAME*	VARCHAR2(15)	NOT NULL	PeopleSoft Record Name
INDEXID*	VARCHAR2(1)	NOT NULL	Identifier of PeopleSoft Index
KEYPOSN*	NUMBER	NOT NULL	Posititon of Field within Key
FIELDNAME	VARCHAR2(100)	NOT NULL	Name of Column, or expression.
ASCDESC	NUMBER	NOT NULL	1=Ascending 0=Descending

Additional Unique Constraint: RECNAME, INDEXID, FIELDNAME

GFC_TEMP_TABLES

This table specifies the PeopleSoft record whose corresponding tables are to be built as Oracle Global Temporary Tables. If a PeopleSoft Temporary Record is specified (rectype=7) then all the instances, both shared and non-shared, will be built as Global Temporary Tables.

Column	**Datatype**	**Null / Constraint**	**Description**
RECNAME*	VARCHAR2(15)	NOT NULL	PeopleSoft Record Name

Working Storage Tables

A number of other tables are used by the package for working storage

GFC_PS_TABLES

This table holds one row for each record to be generated by the package.

Column	Datatype	Null / Constraint	Description
RECNAME*	VARCHAR2(15)	NOT NULL	PeopleSoft Record Name
TABLE_NAME	VARCHAR2(30)	NOT NULL	Database Table name (of shared record if PeopleSoft Temporary Record)

TABLE_TYPE	VARCHAR2(1)		P = Partitioned Table
			
			T = Global Temporary Table

RECTYPE	NUMBER		PeopleSoft Record Type from PSRECDEFN.REC_TYPE 0 = Regular Table 7 = Temporary Table
TEMPTBLINSTANCES	NUMBER		Number of Instances of PeopleSoft Temporary Table
OVERRIDE_SCHEMA	VARCHAR2(30)		Name of Schema in which to build table.
MATCH_DB	VARCHAR2(30)		Y = No difference detected between definition of Table and Partitioning between PeopleSoft and Oracle Catalogues

GFC_PS_TAB_COLUMNS

Column	Datatype	Null / Constraint	Description
RECNAME*	VARCHAR2(15)	NOT NULL	PeopleSoft Record Name
FIELDNAME	VARCHAR2(18)	NOT NULL	PeopleSoft Field Name, which becomes Oracle column name.
USEEDIT	NUMBER	NOT NULL	From PSRECFIELD. Used to define automatically generated Key and Alternate search key indexes in PeopleSoft, and whether DATE and LONG columns are NOT NULL.
FIELDNUM	NUMBER	NOT NULL	Position of field in record, equivalent to position of column in table
SUBRECNAME	VARCHAR2(15)	NOT NULL	Subrecord from which field definition derived[22].

[22] Because the GFC_PSPART package was originally developed on PeopleTools 8.1, it does not rely on PSRECFIELDDB to be accurate, although it is maintained by Application Designer from PeopleTools 8.4. Instead, it works from PSRECFIELD.

GFC_ORA_TAB_COLUMNS

This table is an extract from USER_TAB_COLUMNS of the required tables to improve performance of the package.

Column	Datatype	Null / Constraint	Description
TABLE_NAME*	VARCHAR2(30)	NOT NULL	Database Table Name
COLUMN_NAME*	VARCHAR2(30)	NOT NULL	Database Column Name
COLUMN_ID	NUMBER	NOT NULL	Database Column ID in table

Additional Unique Constraint on TABLE_NAME, COLUMN_ID

GFC_DDL_SCRIPT

This table holds the DDL commands generated by the package

Column	Datatype	Null / Constraint	Description
TYPE	NUMBER	NOT NULL	Script type. Corresponds with p_type parameter on SPOOLER function (see page 55) 0 = gfcbuild script. Similar to a PeopleSoft Alter by Recreation script 1 = gfcindex: Rebuild All indexes 2 = gfcstats: Refresh All statistics 3 = gfcalter: All missing partitions
LINE_NO	NUMBER	NOT NULL	Line Number of Script
LINE	VARCHAR2(4000)		Line of DDL Script

Context

The *gfc_pspart* package uses a system context, also called *gfc_pspart* to persistently hold certain settings.

Context Variable	**Default Value**	**Description**
CHARDEF	N	Use VARCHAR2 character definition.
LOGGING	N	If set to N, DDL script includes NOLOGGING options when building tables and indexes
PARALLEL	Y	Set to Y to include parallel index build option
ROLES	N	Set Y if roles are to be granted to tables
SCRIPTID	GFCBUILD	ID string to be used in name of script and name of project generated.
UPDATE_ALL	NULL	Role to be granted to permanent tables. No role granted if NULL
READ_ALL	NULL	Role to be granted to permanent tables. No role granted if NULL
DROP_INDEX	Y	If set to Y index dropped before rename, else if set to N index is renamed to OLD_
PAUSE	N	Add pause commands to generated script
EXPLICIT_SCHEMA	Y	Explicitly specify PeopleSoft owner ID in DDL script.
BLOCK_SAMPLE	Y	Specify block sampling in update statistics script.
BUILD_STATS	N	If set to Y add gather statistics commands to build script.
DELETETEMPSTATS	Y	If set to Y delete and from Oracle 10g also lock statistics on temporary tables.
LONGTOCLOB	N	If set to Y always create long columns as CLOB, other only do so if PSOPTION.DATABASE_OPTIONS=2
DDLENABLE[23]	NULL	Command to permit DDL to be issued against table

[23] *Changed 5.2.2013:* DDLENABLE and DDLDISABLE replace DDLTRIGGER which previously held the name of the trigger to be disabled and enabled to permit and prevent DDL. The new version of PSFT_DDL_LOCK (see http://www.go-faster.co.uk/scripts.htm#psft_ddl_lock.sql) can be disengaged for just the current session.

DDLDISABLE	NULL	Command to prevent DDL from being isseud against table
DROP_PURGE	Y	Add purge option to DROP TABLE commands on Oracle 10g or higher.
FORCEBUILD	Y	Debug option only – not supported
DESC_INDEX	Y	If N do not include DESC keyword on columns specified in PeopleSoft as descending.
REBUILDDEFLTSUB	N	If Y the default subpartition will exchange out of the table, and the data is copied back into the table when the other partitions have been added. Use this when adding list sub-partitions for which data already exists in the default list sub-partition. If N then the default subpartition having been exchanged out of the table will be exchanged back into it.
REPOPNEWMAX	N	When a new range partition is added to a table that is not the higher partition (which is always the case when a MAXVALUE partition exists) the high partition is split to create the new partition. If Y and a MAXVALUE partition exists, then the MAXVALUE partition is exchanged with a table, the now empty MAXVALUE partition is dropped permitting the new partitions to be added, then a new MAXVALUE partition is added and data is copied back into the table. If there is a lot of data in the MAXVALUE partition this option can be faster. However, data may temporarily disappear from the table. NB: This option does not apply to composite partitioned tables.
DEBUG_LEVEL	0	Positive values enable debug code in the package that is emitted to the terminal. 0=No Debug Code 5=Default debug level. Including beginning of procedure. 8=Report procedure parameters, numbers of rows returned. 7=Explicitly reseting action at end of procedure. 9=Include every line emitted to the script table

Example Meta-Data

Every customer who uses this package has meta-data that is specific to their environment. This is done by providing custom versions of the partdata.sql and other meta-data scripts. *partdata.sql* contains various pieces of sample meta-data that have been commented out as an example of how to define partitioning for additional tables.

Example General Ledger Meta-Data

The partitioning requirements for General Ledger report provide a good example of some of the things that can be done with this utility.

Here the PS_LEDGER table will be range partitions into temporal partitons by range partitioning on the combination of FISCAL_YEAR and ACCOUNTING_PERIOD. There will be month partitions for fiscal year 2006, quarterly for 2005, and annual partitions prior to that.

Comments in the script have been added using footnotes.

```
INSERT INTO gfc_part_tables (recname, part_id, part_column, part_type, subpart_type, hash_partitions,
tab_tablespace, idx_tablespace, tab_storage, idx_storage, stats_type, sample_size, method_opt)
VALUES('LEDGER', 'GL', 'FISCAL_YEAR,ACCOUNTING_PERIOD'[24], 'R', 'LEDGER', 0, 'GLLARGE', 'PSINDEX',
'PCTUSED 90 PCTFREE **PCTFREE**'[25], 'PCTFREE **PCTFREE**', 'Y', NULL, 'FOR ALL COLUMNS SIZE 1', NULL);

INSERT INTO gfc_part_ranges[26] VALUES('GL',2000,' 2000', '2001,0', NULL[27], NULL, 'PCTFREE 0', 'PCTFREE 0'[28]);
INSERT INTO gfc_part_ranges VALUES('GL',2001,2001, '2002,0', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2002,2002, '2003,0', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2003,2003, '2004,0', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2004,2004, '2005,0', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');

INSERT INTO gfc_part_ranges VALUES('GL',2005.0, '2005_BF', '2005,1', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2005.03, '2005_Q1', '2005,4', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2005.06, '2005_Q2', '2005,7', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2005.09, '2005_Q3', '2005,10', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2005.12, '2005_Q4', '2005,999', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2005.99, '2005_CF', '2006,0', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');

INSERT INTO gfc_part_ranges VALUES('GL',2006.0, '2006_BF', '2006,1', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
```

[24] The table will be range partitioned on the combination of FISCAL_YEAR and ACCOUNTING_PERIOD. Two columns, but partitioned in a single dimension.

[25] By using **PCTFREE**, the value for PCTFREE will come from the value specified in the PeopleSoft meta data from the override value on the object if specified, otherwise from the default on the DDL model.

[26] A row is created in GFC_PART_RANGES for each range partition.

[27] Tablespaces have not been specified at range partition level, so all partitions will go into the tablespaces specified at TABLE and INDEX level.

[28] However, the value for PCTFREE has been overridden on historical partitions that hold data for previous years GL data. This data will not be changed, so it is not necessary to preserve free space in each block. This minimises the number of block reads necessary to access the object.

```
INSERT INTO gfc_part_ranges VALUES('GL',2006.01, '2006_M01', '2006,2', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2006.02, '2006_M02', '2006,3', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2006.03, '2006_M03', '2006,4', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2006.04, '2006_M04', '2006,5', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2006.05, '2006_M05', '2006,6', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2006.06, '2006_M06', '2006,7', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2006.07, '2006_M07', '2006,8', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2006.08, '2006_M08', '2006,9', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2006.09, '2006_M09', '2006,10', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2006.10, '2006_M10', '2006,11', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2006.11, '2006_M11', '2006,12', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2006.12, '2006_M12', '2006,999', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
INSERT INTO gfc_part_ranges VALUES('GL',2006.99, '2006_CF', '2007,0', NULL, NULL, 'PCTFREE 0', 'PCTFREE 0');
```

This is an example of how to build a function based index on a table, and then to partition that index, in this case locally partition.

```
-----------------------------------------------------------------------------------------------------------
INSERT data to generate the function based indexed
-----------------------------------------------------------------------------------------------------------
INSERT INTO gfc_ps_idxddlparm (RECNAME,INDEXID,PARMNAME,PARMVALUE)[29]
VALUES ('LEDGER','Z','LEDGER','Z', 1, 0, 0);

INSERT INTO gfc_ps_keydefn VALUES ('LEDGER','Z',1,'FISCAL_YEAR');
INSERT INTO gfc_ps_keydefn VALUES ('LEDGER','Z',2,'ACCOUNTING_PERIOD');
INSERT INTO gfc_ps_keydefn VALUES ('LEDGER','Z',3,'CASE WHEN BUSINESS_UNIT LIKE ''XXX%'' AND LEDGER = ''USXXXX''
THEN ''USXXXX'' ELSE LEDGER END'); [30]
INSERT INTO gfc_ps_keydefn VALUES ('LEDGER','Z',4,'DEPTID');
INSERT INTO gfc_ps_keydefn VALUES ('LEDGER','Z',5,'ACCOUNT');
INSERT INTO gfc_ps_keydefn VALUES ('LEDGER','Z',6,'BUSINESS_UNIT');
```

[29] Additional index data is added to gfc_ps_idxddlparm.

[30] A record is required in gfc_ps_keydefn for each column or expression in the index definition. The expression in the third key is used in many reports, so it makes sense to build a function based index on it, but it still leads on FISCAL_YEAR and ACCOUNTING_PERIOD, and so is also locally partitioned

Example Global Payroll Meta-Data

Some of scripts to build the meta-data for Global Payroll is presented here as an example, with comments and explanations in footnotes. This script is fairly standard for all Global Payroll customers who run their payroll processes streamed.

```
REM gp-partdata.sql
-----------------------------------------------------------------------------------------------------------
--view to identify country extention tables
-----------------------------------------------------------------------------------------------------------
CREATE OR REPLACE VIEW gfc_installed_gp[31] AS
SELECT   recname,
         rectype,
        CASE SUBSTR(r.recname,1,5)
…
             WHEN 'GPGB_' THEN i.INSTALLED_GP_UK
             WHEN 'GPIE_' THEN i.INSTALLED_GP_IRL
…
             ELSE 'Y'
         END AS installed_gp
FROM     psrecdefn r
,        ps_installation i
WHERE   r.rectype IN(0,7) --only SQL tables can be partitioned or rebuilt as GTTs
;
…
```

Most of the temporary tables listed here are in the Payroll Calculation process (GPPDPRUN). If payroll is run streamed multiple concurent processes will share the same tables leading to significant read consistency contention. By converting these tables to Oracle Global Temporary Tables, each session, and therefore each process, will have its own physical copy of each table.

The lists of tables are partly static and partly dynamically generated from PeopleTools and Application meta-data.

```
-----------------------------------------------------------------------------------------------------------
--insert data to describe temporary tables
--country specific tables for installed country extentions only will be added
-----------------------------------------------------------------------------------------------------------
INSERT INTO gfc_temp_tables
SELECT  r.recname
FROM    gfc_installed_gp r
WHERE   r.installed_gp != 'N'
AND     r.rectype IN(0,7) --only normal tables can be partitioned
AND     (r.recname IN( /*payroll calculation work tables*/
                  'GP_CAL_TS_WRK',
                  'GP_CANC_WRK',
                  'GP_CANCEL_WRK',   /*new in 8.4*/
```

[31] This view is used to work out which Global Payroll country extentions are installed. Only country specific result tables for installed country extentions will be partititioned or converted to Global Temporary Tables.

```
                    'GP_DB2_SEG_WRK',  /*new in 8.4*/
                    'GP_DEL_WRK',
                    'GP_DEL2_WRK',     /*new in 8.4*/
                    'GP_EXCL_WRK',
                    'GP_FREEZE_WRK',   /*new in 8.4*/
                    'GP_HST_WRK',
                    'GP_JOB2_WRK',     /*13.2.2008*/
                    'GP_MSG1_WRK'    ,'GP_MSG3_WRK',
                    'GP_NEW_RTO_WRK' ,
                    'GP_OLD_RTO_WRK' ,
                    'GP_PI_HDR_WRK',
                    'GP_PYE_HIST_WRK', /*13.2.2008*/
                    'GP_PYE_HIS2_WRK', /*13.2.2008*/
                    'GP_PYE_ITER_WRK', /*new in 8.4*/
                    'GP_PYE_ITR2_WRK', /*new in 8.4*/
                    'GP_PYE_RCLC_WRK', /*new in hcm9.0 - added 21.5.2008*/
                    'GP_PYE_STAT_WRK',
                    'GP_PYE_STA2_WRK', /*13.2.2008*/
                    'GP_RTO_CRD_WRK', /*20.2.2008*/
                    'GP_RTO_PRC_WRK' ,
                    'GP_RTO_TRG_WRK1', /*new in 8.4*/
                    'GP_RTO_TRGR_WRK',
                    'GP_SEG_WRK',
                    'GP_TLPTM_WRK',
                    'GP_TLSNT_WRK',
                    'GP_TLTRC_WRK',
                    'GP_TL_PIGEN_WRK',
                    'GP_TL_PIHDR_WRK',
                    'GP_TL_TRG_WRK',
                    /*pin packager*/
                    'GP_PKG_ELEM_WRK',
                    /*GL*/
                    'GP_ACC_LINE_STG', --added 5.3.2009
                    'GP_GL_AMT1_TMP' ,'GP_GL_AMT2_TMP',
                    'GP_GL_DATA_TMP' ,
                    'GP_GL_DNF_TMP'  ,
                    'GP_GL_MAPI_TMP' ,
                    'GP_GL_OLD_TMP'  ,
                    'GP_GL_SEG_TMP'  ,
                    'GP_GL_SEGV_TMP' ,
                    'GP_GL_STO6_TMP' ,
                    'GP_GL_S7N8_TMP' ,
                    /*Banking*/
                    'GP_NET_PAY1_TMP','GP_NET_PAY2_TMP','GP_NET_PAY3_TMP',
                    'GP_NET_DST1_TMP','GP_NET_DST2_TMP',
                    'GP_PAYMENT_TMP' ,'GP_PAYMENT2_TMP',
                    'GP_REV_DLTA_TMP', /*added 9.6.2004*/
                    'GP_SRC_BNK1_TMP','GP_SRC_BNK2_TMP',
                    /*Swiss Banking*/
                    'GPCH_BK_TMP1','GPCH_BK_TMP2',
                    'GPCH_BK_PMTTYPE',
                    /*reporting*/
                    'GPCH_BL_PRINT_T',
                    'GPCH_BL_PRT',
                    'GPCH_RP_AL1',
                    'GPCH_RP_AL01',
                    'GPCH_RP_AL03'  , 'GPCH_RP_AL03_1',
                    'GPCH_RP_AL07_1', 'GPCH_RP_AL07_2', 'GPCH_RP_AL07_3',
                    'GPCH_RP_AL08',
                    'GPCH_RP_AL81'  , 'GPCH_RP_AL82'   , 'GPCH_RP_AL83',
                    'GPCH_RP_TX01A' ,
                    'GPCH_RP_TX06'  , 'GPCH_RP_TX06_01',
                    'GPCH_RP_TX61'  , 'GPCH_RP_TX62'   , 'GPCH_RP_TX63',
                    'GPCH_RP_FK1A','GPCH_RP_FK2A',
                    'GPCH_RP_0001_01',
                    'GPCH_SRC_BNK',
                    'GPCHAL021_TMP','GPCHAL022_TMP','GPCHAL023_TMP',
                    'GPCHAL024_TMP',
                    'GPCHAL031_TMP',
                    'GPCHAL051_TMP','GPCHAL052_TMP',
                    'GPCHAL071_TMP','GPCHAL072_TMP','GPCHAL073_TMP','GPCHAL074_TMP','GPCHAL075_TMP',
                    'GPCHAL101_TMP','GPCHAL102_TMP',
                    'GPCHSI061_TMP',
                    'GPCHST021_TMP','GPCHST022_TMP','GPCHST023_TMP',
                    'GPCHTX011_TMP','GPCHTX012_TMP',
                    'GPCHTX021_TMP',
                    'GPCHTX061_TMP','GPCHTX062_TMP','GPCHTX063_TMP','GPCHTX064_TMP',
                    'GPGB_PSLIP_ED_D','GPGB_PSLIP_BL_D', /*gpgb_pslip can now be run stream 4.2.2004*/
                    'GPGB_PSLIP_ED_W','GPGB_PSLIP_BL_W',
                    --customer tables
…
                    )
OR r.recname IN(
                    SELECT t.recname[32]
                    FROM   psaeappltemptbl t
                    ,      psaeappldefn a
                    WHERE  a.ae_applid = t.ae_applid
                    AND    a.ae_disable_restart = 'Y' --restart is disabled
                     AND    a.ae_applid IN('GP_PMT_PREP','GP_GL_PREP'
                                   ,'GPGB_PSLIP','GPGB_PSLIP_X', 'GPGB_EDI') /*limited to GP AE processes*/
                    ))
/
```

[32] This subquery adds any records that are used as temporary working storage tables in the GP Application Engine programs that can be streamed and that have restart disabled.

```
-----------------------------------------------------------------------------------------------------------
--insert data to specify the tables to be partitioned
--country specific tables for installed country extentions only will be added
-----------------------------------------------------------------------------------------------------------
DELETE FROM gfc_part_tables
WHERE part_id = 'GP'
/

INSERT INTO gfc_part_tables
(recname, part_id, part_column, part_type)
SELECT  r.recname, 'GP'
,        'EMPLID', 'R'
FROM    gfc_installed_gp r
WHERE   r.installed_gp != 'N'
AND      r.rectype = 0 --only normal tables can be partitioned
AND     (       r.recname IN(
                        'GP_AUDIT_TBL',           /*added 21.5.2008*/
                        'GP_ABS_EVENT',           /*absence - added 3.10.2003*/
                        'GP_GL_DATA',             /*gl transfer table*/
                        'GP_GRP_LIST_RUN',        /*new in 8.4*/
                        'GP_ITER_TRGR',
                        'GP_MESSAGES',
                        'GP_PAYMENT',
                        'GP_PI_GEN_HDR',          /*13.2.2008-added*/
                        'GP_PI_GEN_DATA',         /*postitive input*/
                        'GP_PI_GEN_REF',          /*13.2.2008-added for kelly-TL postitive input*/
                        'GP_PI_GEN_SOVR',         /*14.11.2008-added*/
                        'GP_PI_MNL_DATA',         /*postitive input*/
                        'GP_PI_MNL_SOVR',         /*postitive input*/
                        'GP_PYE_ITER_LST',        /*13.2.2008-added*/
                        'GP_PYE_OVRD',
                        'GP_PYE_PRC_STAT',
                        'GP_PYE_SEG_STAT',
                        'GP_RCP_PYE_DTL',         /*added 7.6.2004 for gp_pmt_prep*/
                        'GP_RSLT_ABS',            /*payroll calculation results*/
                        'GP_RSLT_ACUM',           /*payroll calculation results*/
                        'GP_RSLT_DELTA',          /*payroll calculation results*/
                        'GP_RSLT_ERN_DED',        /*payroll calculation results*/
                        'GP_RSLT_PI_DATA',        /*payroll calculation results*/
                        'GP_RSLT_PI_SOVR',        /*payroll calculation results*/
                        'GP_RSLT_PIN',            /*payroll calculation results*/
                        'GP_RTO_TRG_CTRY',        /*8.1 + 8.3 only*/
                        'GP_RTO_TRGR',            /*deadlock problem*/
                        'GPCH_BK_XFER_EE',        /*bank transfer*/
                        'GPCH_TX_DATA',           /*tax data table - added 19.6.2003 - to improve scan and reduce
latch contention*/
                        'GPGB_ABS_EVT_JR',
                        'JOB',                    /*hr data */
                        'COMPENSATION',           /*hr compensation data added 18.1.2010*/
                        'X_PYE_OVRD_ET',          /*customer table*/
                        'GPGB_PAYMENT',           /*added 12.3.2004*/
                        'GPGB_PSLIP_P_ED',        /*uk payslip process gpgb_pslip can now be run streamed*/
                        'GPGB_PSLIP_P_BL',        /*uk payslip process gpgb_pslip can now be run streamed*/
                        'GPGB_PSLIP_P_HR',        /*uk payslip process gpgb_pslip can now be run streamed*/
                        'GPGB_PSLIP_P_FT',        /*uk payslip process gpgb_pslip can now be run streamed*/
```

```
                           'TL_PAYABLE_TIME'        /*13.2.2008-added-TL*/
                        )
        OR      r.recname IN(                       /*range partition any writable arrays*/
                           SELECT  recname
                        FROM    ps_gp_wa_array[33]
                        )
        )
/
```

Sometimes, the easiest way to remove archived data from the payroll tables is to rebuild them, and exclude the unwanted data. A criterion can be specified for each table, and it gets added to INSERT ... SELECT command. Usually, this is left blank

```
-----------------------------------------------------------------------------------------------------------
--process the filter column here
-----------------------------------------------------------------------------------------------------------
UPDATE gfc_part_tables p
SET    p.criteria = ''
/

UPDATE gfc_part_tables p
SET    p.criteria = 'WHERE cal_run_id >= ''XX2008'''
WHERE  EXISTS(
                  SELECT   'x'
                  FROM     psrecfield f
                  WHERE    p.recname = f.recname
                  and      f.fieldname IN('CAL_RUN_ID', 'SRC_CAL_RUN_ID', 'CAL_ID')
                  )
AND p.criteria IS NULL
AND    1=2
;

UPDATE gfc_part_tables p
SET    p.criteria = ''
WHERE  p.recname IN('JOB','GP_ABS_EVENT','GPGB_ABS_EVT_JR')
;

ttitle 'Filter Conditions'
column recname format a18
column criteria format a60
SELECT   recname, criteria
FROM     gfc_part_tables
ORDER BY 1
/
ttitle off
```

[33] In Global Payroll some results can be written to what PeopleSoft calls ‘writable arrays’. These are separate database tables, and so should also be partitioned.

The two largest GP result tables are good candidates for list sub-partitioning by calendar group ID. Sometimes, other tables are possible candidates.

```
-----------------------------------------------------------------------------------------------------------
--specify list subpartitioned tables
-----------------------------------------------------------------------------------------------------------
UPDATE   gfc_part_tables
SET      subpart_type = 'L'
,        subpart_column = 'CAL_RUN_ID'
,        hash_partitions = 0
WHERE    recname IN('GP_RSLT_ACUM', 'GP_RSLT_PIN'
--,'GP_GL_DATA' --not always worth subpartitioning
--,'GP_PYE_SEG_STAT' -- subpartitioning does not always work well with retro queries
--,'GP_PYE_PRC_STAT' -- subpartitioning does not always work well with retro queries
--,'GP_RSLT_PI_SOVR', 'GP_RSLT_PI_DATA' --14.2.2008 removed
)
/
```

The storage options can be set keep only a small amount of free space on most of the result tables. However, experience has shown that GP_PYE_SEG_STAT needs about 15% free space because it is updated during the payroll calculation with status information.

```
-----------------------------------------------------------------------------------------------------------
--set storage options on partitioned objects
-----------------------------------------------------------------------------------------------------------
UPDATE   gfc_part_tables
SET      tab_tablespace = 'GPAPP'   /*default PSFT tablespace*/
,        idx_tablespace = 'PSINDEX' /*default PSFT tablespace*/
,        tab_storage = 'PCTUSED 95 PCTFREE 1'
,        idx_storage = 'PCTFREE 1'
/


UPDATE   gfc_part_tables
SET      tab_storage = 'PCTUSED 80 PCTFREE 15'
WHERE    recname IN('GP_PYE_SEG_STAT')
/
```

Global Indexes

Not all indexes should be locally partitioned. This statement generates meta-data to suppress partitioning of the index if the partitioning key does not appear in the first three columns of the index. The result is a non-partitioned global index.

```
-----------------------------------------------------------------------------------------------------------
--describe indexes that are not to be locally partitioned
-----------------------------------------------------------------------------------------------------------
INSERT INTO gfc_part_indexes
(recname, indexid, part_id, part_type, part_column, subpart_type, subpart_column, hash_partitions)
SELECT   t.recname
,        i.indexid
,        t.part_id
,        t.part_type
,        t.part_column
,        'N' subpart_type
,        '' subpart_column
,        t.hash_partitions
FROM     gfc_part_tables t
,        psindexdefn i
WHERE    t.recname = i.recname
AND      t.subpart_type = 'L'
AND NOT EXISTS(
         SELECT 'x'
         FROM pskeydefn k, psrecfielddb f
         WHERE f.recname = i.recname
         AND   k.recname = f.recname_parent
         AND   k.indexid = i.indexid
         AND   k.fieldname = t.subpart_column
         AND   k.keyposn <= 3
         )
;
```

This is another example of an index that should not be partitioned. The meta-data is explicitly inserted.

```
INSERT INTO gfc_part_indexes
(recname, indexid, part_id, part_type, part_column)
VALUES
('TL_PAYABLE_TIME','A','GP','N',' ')
/
```

Global Payroll Streams

In Global Payroll, the 'Streams' are defined as ranges of EMPLID in the table PS_GP_STRM. The meta-data that describes the range partitions for the payroll tables is derived directly from this table. Thus, once the streams have been defined, the partitioning will automatically match the stream definition, and there will be a 1:1 mapped of payroll process to physical table partition. Thus each block of each partition cannot be updated by more than one process at a time, and there is no possibility of consistent read having to be done by the database in payroll processes. The result is very good scalability of payroll with the number of streams.

```
-----------------------------------------------------------------------------------------------------------
--insert data to specify range partitioning strategry
-----------------------------------------------------------------------------------------------------------
DELETE FROM gfc_part_ranges
WHERE part_id = 'GP'
/

INSERT INTO gfc_part_ranges
(part_id, part_no, part_name, part_value)
SELECT   'GP', strm_num
,        LTRIM(TO_CHAR(strm_num,'000')) part_name
,        NVL(LEAD(''''||emplid_from||'''',1) OVER (ORDER BY strm_Num),'MAXVALUE') part_value
FROM     ps_gp_strm
/


UPDATE   gfc_part_ranges[34]
SET      tab_tablespace = 'GPSTRM'||LTRIM(TO_CHAR(MOD(part_no-1,32)+1,'00'))||'TAB'
,        idx_tablespace = 'GPSTRM'||LTRIM(TO_CHAR(MOD(part_no-1,32)+1,'00'))||'IDX'
--SET    tab_tablespace = 'GPSTRM'||part_name||'TAB'
--,      idx_tablespace = 'GPSTRM'||part_name||'IDX'
--SET    tab_tablespace = 'GPTABPART'||part_name||''
--,      idx_tablespace = 'GPIDXPART'||part_name||''
--,      tab_storage = '/*TAB STORAGE*/'
--,      idx_storage = '/*IDX STORAGE*/'
WHERE 1=1
/
```

[34] A pair of tablespaces is created for each payroll stream. All the tables for the same stream go into the same tablespace. There is no partiticular performance need for this, but it helps to determine how much I/O came from which stream.

Function Based Indexes

Sometimes, it will be necessary to build indexes that are not defined in PeopleSoft. A typical example is a function based index which cannot be defined in PeopleSoft.

```
-----------------------------------------------------------------------------------------------------------
--insert data to generate the function based indexed
-----------------------------------------------------------------------------------------------------------
--(recname,indexid,keyposn,fieldname)
INSERT INTO gfc_part_indexes (recname, indexid, part_id, part_type, part_column, idx_storage, name_suffix)
VALUES ('TL_PAYABLE_TIME','Y','GP','N',' ', 'PCTFREE 1','_SPARSE');[35]
INSERT INTO gfc_ps_indexdefn (recname, indexid, subrecname, subindexid, platform_ora, custkeyorder, uniqueflag)
VALUES ('TL_PAYABLE_TIME','Y','TL_PAYABLE_TIME','Y', 1, 0, 0);

--(recname,indexid,keyposn,fieldname)
INSERT INTO gfc_ps_keydefn
VALUES ('TL_PAYABLE_TIME','Y',1,'DECODE(PAYABLE_STATUS,''SP'',''SP'',NULL)',1);
INSERT INTO gfc_ps_keydefn
VALUES ('TL_PAYABLE_TIME','Y',2,'DECODE(PAYABLE_STATUS,''SP'',EMPLID,NULL)',1);
INSERT INTO gfc_ps_keydefn
VALUES ('TL_PAYABLE_TIME','Y',3,'DECODE(PAYABLE_STATUS,''SP'',DUR,NULL)',1);
INSERT INTO gfc_ps_keydefn
VALUES ('TL_PAYABLE_TIME','Y',4,'DECODE(PAYABLE_STATUS,''SP'',EMPL_RCD,NULL)',1);

INSERT INTO gfc_part_indexes (recname, indexid, part_id, part_type, part_column, idx_storage, name_suffix)
VALUES ('TL_PAYABLE_TIME','Z','GP','L',' ', 'PCTFREE 1','_SPARSE');[36]
INSERT INTO gfc_ps_indexdefn (recname, indexid, subrecname, subindexid, platform_ora, custkeyorder, uniqueflag)
VALUES ('TL_PAYABLE_TIME','Z','TL_PAYABLE_TIME','Z', 1, 0, 0);

INSERT INTO gfc_ps_keydefn
VALUES ('TL_PAYABLE_TIME','Z',1,'DECODE(PAYABLE_STATUS,''NA'',''NA'',NULL)',1);
INSERT INTO gfc_ps_keydefn
VALUES ('TL_PAYABLE_TIME','Z',2,'DECODE(PAYABLE_STATUS,''NA'',EMPLID,NULL)',1);
INSERT INTO gfc_ps_keydefn
VALUES ('TL_PAYABLE_TIME','Z',3,'DECODE(PAYABLE_STATUS,''NA'',DUR,NULL)',1);
INSERT INTO gfc_ps_keydefn
VALUES ('TL_PAYABLE_TIME','Z',4,'DECODE(PAYABLE_STATUS,''NA'',EMPL_RCD,NULL)',1);
```

[35] Index PSYTL_PAYABLE_TIME_SPARE will be created on table PS_TL_PAYABLE_TIME. The suffix is added to the index name to eliminate the risk that PeopleTools will attempt to drop it. This index will be a Global non-partitioned index.

[36] Index PSZTL_PAYABLE_TIME_SPARE will also be created on table PS_TL_PAYABLE_TIME. This index will be locally partitioned index.

This is the output generated

```
CREATE INDEX sysadm.psytl_payable_time_sparse ON sysadm.ps_tl_payable_time
(DECODE(PAYABLE_STATUS,'SP','SP',NULL)
,DECODE(PAYABLE_STATUS,'SP',EMPLID,NULL)
,DECODE(PAYABLE_STATUS,'SP',DUR,NULL)
,DECODE(PAYABLE_STATUS,'SP',EMPL_RCD,NULL)
)
TABLESPACE PSINDEX
…
CREATE INDEX sysadm.psztl_payable_time_sparse ON sysadm.ps_tl_payable_time
(DECODE(PAYABLE_STATUS,'NA','NA',NULL)
,DECODE(PAYABLE_STATUS,'NA',EMPLID,NULL)
,DECODE(PAYABLE_STATUS,'NA',DUR,NULL)
,DECODE(PAYABLE_STATUS,'NA',EMPL_RCD,NULL)
)
LOCAL
(PARTITION tl_payable_timez2008l09 TABLESPACE GP2008L09IDX
…
,PARTITION tl_payable_timez2010l13 TABLESPACE GP2010L13IDX
,PARTITION tl_payable_timezz_others
)
TABLESPACE PSINDEX
…
```

List Partitioning

The requirements for list partitions vary greatlty from customer to customer. The insert statements do extract the calendar group names from the application tables because the calendars might not have been created at the time when the DBA needs to create the partitions.

Again, the following are examples collected from several sites.

```
-----------------------------------------------------------------------------------------------------------
--insert data to list partitions
--2007 onwards
-----------------------------------------------------------------------------------------------------------
DELETE FROM gfc_part_lists
WHERE part_id = 'GP'
/
INSERT INTO gfc_part_lists[37]
(part_id, part_no, part_name, list_value)
VALUES ('GP',9999,'Z_OTHERS','DEFAULT')
/

-----------------------------------------------------------------------------------------------------------
--monthly partitions for Pensioners[38]
-----------------------------------------------------------------------------------------------------------
INSERT INTO gfc_part_lists
(part_id, part_no, part_name, list_value)
SELECT   'GP'
,        year
,              LTRIM(TO_CHAR(y.year,'0000'))
,        ''''||LTRIM(TO_CHAR(y.year,'0000'))||'PM01'','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'PM02'','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'PM03'','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'PM04'','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'PM05'','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'PM06'','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'PM07'','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'PM08'','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'PM09'','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'PM10'','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'PM11'','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'PM12'''
FROM (
         SELECT   2007+rownum as year[39]
         FROM     dba_objects
         WHERE    rownum <= 3
         ) y
```

[37] There should always be a DEFAULT list partition. And rows for which the value of the list partitioning column does not match any other partition will go in here. If this was omitted, there would be a risk of an insert or update operation failing with an error, and crashing a process.

[38] One list partition per year. Each partition contains 12 calendar group IDs.

[39] Partitions will be created for 3 tax years starting with 2008-09.

```
ORDER BY 1,2,3
/
```

```
-----------------------------------------------------------------------------------------------------------
--lunar monthly partitions for lunar and weekly[40]
-----------------------------------------------------------------------------------------------------------
INSERT INTO gfc_part_lists
(part_id, part_no, part_name, list_value)
SELECT   'GP'
,        year+period/100
,        LTRIM(TO_CHAR(y.year,'0000'))||'L'||LTRIM(TO_CHAR(p.period,'00'))
,        ''''||LTRIM(TO_CHAR(y.year,'0000'))||'UL'||LTRIM(TO_CHAR(  p.period  ,'00'))||''','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'UW'||LTRIM(TO_CHAR(4*p.period-3,'00'))||''','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'UW'||LTRIM(TO_CHAR(4*p.period-2,'00'))||''','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'UW'||LTRIM(TO_CHAR(4*p.period-1,'00'))||''','||
         ''''||LTRIM(TO_CHAR(y.year,'0000'))||'UW'||LTRIM(TO_CHAR(4*p.period-0,'00'))||''''
FROM     (
         SELECT  rownum as period
         FROM     dba_objects
         WHERE    rownum <= 14
         ) p
,        (
         SELECT   2007+rownum as year
         FROM     dba_objects
         WHERE    rownum <= 3 --years of list partitions
         ) y
WHERE    period <= DECODE(y.year,2023,14,13)
ORDER BY 1,2,3
/

--need to add specific years where W53
UPDATE   gfc_part_lists a[41]
SET      a.list_value = a.list_value ||','''||SUBSTR(a.part_name,1,4)||'UW53'''
WHERE    a.part_id = 'GP'
AND      a.part_name = (
                  SELECT   MAX(b.part_name)
                  FROM     gfc_part_lists b
                  WHERE    a.part_id = 'GP'
                  AND      SUBSTR(a.part_name,1,5) = SUBSTR(b.part_name,1,5))
AND      (a.part_name LIKE '2013_' OR
         a.part_name LIKE '2023_') -- and others
/
```

[40] Here, there will be 13 partitions per year, 14 in a lunar leap year. Each period will contain one period for lunar pay cycles, the 4 weekly calendar groups for the same period, and

[41] The 53rd weekly period will be added to the 13th partition for that year.

Selectively Building Sub-Partitions

It is not always necessary to build every list partition for every range partition.

```
---------------------------------------------------------------------------------------------------------
--mapping between ranges and lists
---------------------------------------------------------------------------------------------------------
DELETE FROM gfc_part_range_lists
WHERE part_id = 'GP'
/

INSERT INTO gfc_part_range_Lists[42]
(part_id, range_name, list_name)
SELECT r.part_id, r.part_name, l.part_name
FROM   gfc_part_ranges r
,      gfc_part_lists l
WHERE  l.part_id = r.part_id
/

---------------------------------------------------------------------------------------------------------
--delete range/list combinations that are not needed
---------------------------------------------------------------------------------------------------------
DELETE FROM gfc_part_range_lists
WHERE build = 'Y'
AND   (  (list_name like 'IRL%' AND range_name != '01')[43]
      OR (list_name like 'UK%'  AND NOT range_name IN('02','03','04','05','06','07','08')))[44]
AND   build = 'Y'
AND   part_id = 'GP'
/

--Uncomment this if you to just rebuild composite partitioned tables
--DELETE FROM gfc_temp_tables --WHERE RECNAME != ' '
--;
--DELETE FROM gfc_part_tables
--WHERE subpart_type = 'N'
--RECNAME != 'GP_PYE_SEG_STAT'
--;
```

[42] For ease of processing a table with ever combination of range and list is built, and then unwanted partitions will be marked.

[43] Here the EMPLIDs of a company's Irish employees always fell into stream 1. So we don't want to build the list partitions for the Irish Calendar Group IDs in any stream other than stream 1. If we did they would be empty. The Irish payroll is not streamed, but it is still advantageous to have separate list sub-partitions for each period.

[44] The UK employees are in streams 2 through 8, and not in stream 1, so there is no point having list partitions for UK calendar group IDs in stream 1.

Setting Context Values

The final part of the meta-data is to set site specific settings in the context via another public procedure in the *gfc_pspart* package (see SET_DEFAULTS Procedure on page 52).

```
------------------------------------------------------------------------------------------------------------
--set gfc_pspart defaults
------------------------------------------------------------------------------------------------------------
set serveroutput on buffer 1000000000
execute gfc_pspart.set_defaults(p_read_all=>'XXX_READ_ONLY');
execute gfc_pspart.set_defaults(p_update_all=>'XXX_UPDATE');
execute gfc_pspart.set_defaults(p_roles => 'Y');
execute gfc_pspart.set_defaults(p_ddlenable=>'BEGIN psft_ddl_lock.set_ddl_permitted(TRUE); END;'||CHR(10)||'/');[45]
execute gfc_pspart.set_defaults(p_ddldisable=>'BEGIN psft_ddl_lock.set_ddl_permitted(FALSE); END;' ||CHR(10)||'/');
execute gfc_pspart.display_defaults;
```

Globally Partitioned Indexes

Globally partitioned indexes must have a MAXVALUE partition specified. So in this case the range partition definition for TL is copied to TLMAX, and a MAXVALUE partition is added.

```
--TLMAX is identical to TL but has maxvalue to support globally partitioned indexes
INSERT INTO gfc_part_ranges
(part_id, part_no, part_name, part_value, tab_tablespace, idx_tablespace)
SELECT   'TLMAX', part_no, part_name, part_value, tab_tablespace, idx_tablespace
FROM     gfc_part_ranges
WHERE    part_id = 'TL'
/

INSERT INTO gfc_part_ranges
(part_id, part_no, part_name, part_value)
VALUES
('TLMAX', 9999, 9999, 'MAXVALUE')
/
```

[45] If you still want to disable the entire trigger, the put the ALTER TRIGGER command in these settings.

```
execute gfc_pspart.set_defaults(p_ddlenable  => 'ALTER TRIGGER psft_ddl_lock DISABLE'||CHR(10)||'/');
execute gfc_pspart.set_defaults(p_ddldisable => 'ALTER TRIGGER psft_ddl_lock ENABLE'||CHR(10)||'/');
```

Example Time & Labor Meta-Data

Archiving Time-Based Range Partitioning

```
UPDATE   gfc_part_tables
SET      arch_flag = 'A'[46]
,        arch_schema = 'PSARCH'[47]
WHERE    part_id IN('SCH')
OR       recname IN('AUDIT_SCH_TBL')
/

UPDATE   gfc_part_tables
SET      noarch_condition = 'sch_adhoc_ind = ''1'''[48]
WHERE    recname IN('SCH_DEFN_TBL','SCH_DEFN_DTL','SCH_DEFN_ROTATN','SCH_DEFN_SHFT')
/
```

```
INSERT INTO gfc_part_ranges
(         part_id,   part_no,   part_name,   part_value,   tab_tablespace,    idx_tablespace,    arch_flag)
SELECT   y.part_id, y.part_no, y.part_name, y.part_value, t.tablespace_name, i.tablespace_name, y.arch_flag
FROM     (
         SELECT   'SCH' part_id
         ,        TO_NUMBER(TO_CHAR(mydate,'iyIW')) part_no
         ,        TO_CHAR(mydate,'iyIW') part_name
         ,        'TO_DATE('''||TO_CHAR(MAX(mydate)+1,'YYYYMMDD')||''',''YYYYMMDD'')' part_value
         ,        'TL'||TO_CHAR(MAX(mydate),'YYYY')||'M'||TO_CHAR(MAX(mydate),'MM')||'TAB' tab_tablespace
         ,        'TL'||TO_CHAR(MAX(mydate),'YYYY')||'M'||TO_CHAR(MAX(mydate),'MM')||'IDX' idx_tablespace
         ,        CASE WHEN MAX(mydate)+1<ADD_MONTHS(SYSDATE,-12) THEN 'A' ELSE 'N' END arch_flag --[49]archiving
--       ,        CASE WHEN MAX(mydate)+1<TO_DATE('20100101','yyyymmdd') THEN 'A' ELSE 'N' END arch_flag --
archiving
--       ,        CASE WHEN MAX(mydate)+1<TO_DATE('20090101','yyyymmdd') THEN 'A' ELSE 'N' END arch_flag --
archiving
         FROM     (
                  SELECT   a.from_dt+b.n mydate
                  from     (
                           select TO_DATE('01102008','DDMMYYYY') from_dt
                           from dual
```

[46] Tables to be archived or purged by dropping partitions can be identified in the meta data.

[47] In this case the data will be moved to an archive table with the same name but in a different schema.

[48] Sometimes it is necessary to preserve data that would otherwise be removed by partition exchange. The NOARCH_CONDITION column can be used to hold the logical criteria to select data to be retained. GFC_PSPART does not use this directly, but the partition exchange package also uses the GFC_PSPART meta-data.

[49] The archive flag on GFC_PART_RANGES indicates which range partitions should be archived. Note that partitions will only be archived if both the GFC_PART_TABLES.ARCH_FLAG and GFC_PART_RANGES.ARCH_FLAG are set.

```
                            ) a
                    ,       (
                            select rownum n
                            from dual
                            connect by level <= (SYSDATE-TO_DATE('20081107','yyyymmdd'))
                            ) b
                    ) x
         WHERE mydate >= TO_DATE('20081107','yyyymmdd')
         GROUP BY TO_CHAR(mydate,'iyIW')
         HAVING MIN(mydate) < TO_DATE('20120501','yyyymmdd')
         ) y
         left outer join dba_tablespaces t on t.tablespace_name = y.tab_tablespace
         left outer join dba_tablespaces i on i.tablespace_name = y.idx_tablespace
ORDER BY 1,2,3
/
```

Meta-Data Report

Part of the partdata.sql script generates a report of the meta-data that contains a number of sections

Partitioned Tables

```
Thu Nov 26                                                                                         page    1
                                              Partitioned Tables

PeopleSoft                 Part       Part SubP Sub-Part        Hash Table                Index
Record Name        Part ID Column     Type Type Column         Parts TblSpc               TblSpc
------------------ ------- ---------- ---- ---- -------------- ----- -------------------- --------------------
Table                Index                Stats Sample Optimizati Override
Storage Clause       Storage Clause       Opt   Size % Method     Schema
-------------------- -------------------- ----- ------ ---------- ----------
Criteria
--------------------------------------------------------------------------------
…
GP_RSLT_ACUM       GP      EMPLID     R    L    CAL_RUN_ID         0 GPAPP                PSINDEX
PCTUSED 95 PCTFREE 1 PCTFREE 1            Y
…
SCH_ADHOC_DTL      TL      DUR        R    N                       0 TLLARGE              PSINDEX
INITRANS 4 PCTUSED 9 INITRANS 4 PCTFREE 0 Y
0 PCTFREE 1
…
```

Range Partitioning

```
Thu Nov 26                                                                                           page    1
                                              Range Partitioning

            Part Part            Part                           Table                Index
Part ID      No. Name            Value                          TblSpc               TblSpc
------- -------- --------------- ------------------------------ -------------------- --------------------
Table                Index
Storage Clause       Storage Clause
-------------------- --------------------
GP           1.00 001            'KUL451'                       GPSTRM01TAB          GPSTRM01IDX
GP           2.00 002            'KUL452'                       GPSTRM02TAB          GPSTRM02IDX
GP           3.00 003            'KUL453'                       GPSTRM03TAB          GPSTRM03IDX
GP           4.00 004            'KUL454'                       GPSTRM04TAB          GPSTRM04IDX
…
TL        1218.00 1218           TO_DATE('20120507','YYYYMMDD')
…
TLMAX     1218.00 1218           TO_DATE('20120507','YYYYMMDD')
TLMAX     9999.00 9999           MAXVALUE
```

List Partitioning

```
Thu Nov 26                                                                                          page    1
                                              List Partitioning


            Part Part
Part ID      No. Name
------- -------- ---------------
List                                                                                    Table
Value                                                                                   TblSpc
-------------------------------------------------------------------------------------- --------------------
Index                Table                Index
TblSpc               Storage Clause       Storage Clause
-------------------- -------------------- --------------------
GP       2008.01 2008L01
'2008GL01','2008GW01','2008GW02','2008GW03','2008GW04','2008UL01','2008UW01','2008UW0 GP2008L01TAB
2','2008UW03','2008UW04','2008AA01','2008AA02','2008AA03'
GP2008L01IDX

GP       2008.02 2008L02
'2008GL02','2008GW05','2008GW06','2008GW07','2008GW08','2008UL02','2008UW05','2008UW0 GP2008L02TAB
6','2008UW07','2008UW08','2008AA05','2008AA06','2008AA07'
GP2008L02IDX
…
```

Global and Globally Partitioned Indexes

```
Thu Nov 26                                                                                          page    1
                                      Non-Locally Partitioned Indexes

PeopleSoft                       Part       Part SubP Sub-Part        Hash Index
Record Name        IND Part ID Column     Type Type Column         Parts TblSpc
------------------ --- ------- ---------- ---- ---- -------------- ----- --------------------
Index                Override
Storage Clause       Schema
-------------------- ----------
TL_PAYABLE_TIME    A   GP                 N    N


TL_PAYABLE_TIME    C   GP                 N    N


TL_PAYABLE_TIME    D   GP                 N    N


TL_PAYABLE_TIME    E   GP                 N    N


TL_PAYABLE_TIME    F   TLMAX   DUR        R    N
COMPRESS 1[50]
```

[50] Note that globally partitioned indexes must have an unlimited upper range, so the last value must use a MAXVALUE operator.

Specified Tablespaces

It is useful to be able to tell the DBAs which tablespaces are required.

```
Thu Nov 26
page    1
                                            Specified Tablespaces


    Table                Index
PAR TblSpc               TblSpc
--- -------------------- --------------------
    GPAPP                PSINDEX
    TLAPP                PSINDEX
    TLLARGE              PSINDEX
…
L   GP2008L01TAB         GP2008L01IDX
L   GP2008L02TAB         GP2008L02IDX
…
R   GPSTRM01TAB          GPSTRM01IDX
R   GPSTRM02TAB          GPSTRM02IDX
R   GPSTRM03TAB          GPSTRM03IDX
R   GPSTRM04TAB          GPSTRM04IDX
…
```

This section lists the tablespaces that are referenced in the meta data, and that have not beein built.

```
Thu Nov 26                                                                                     page    1
                                          Tablespaces to be created


Table
TblSpc
--------------------
GP2008L01IDX
GP2008L01TAB
GP2008L02IDX
GP2008L02TAB
```

Package GFC_PSPART

The GFC_PSPART contains a number of public procedures that can be called.

BANNER Procedure

This procedure prints the copyright message.

Syntax

```
gfc_pspart.banner;
```

There are no parameters to this procedure.

```
GFC_PSPART - Partitioned/Global Temporary Table DDL generator for PeopleSoft
(c)Go-Faster Consultancy Ltd. www.go-faster.co.uk 2001-2009
```

NB: This procedure outputs via DBMS_OUTPUT. You will need to *set serveroutput on* in SQL*Plus to obtain any output.

DISPLAY_DEFAULTS Procedure

This procedure prints the value of all the parameters in the system context (see Context on page 26).

Syntax

```
gfc_pspart.display_defaults;
```

There are no parameters to this procedure.

```
GFC_PSPART - Partitioned/Global Temporary Table DDL generator for PeopleSoft
(c)Go-Faster Consultancy Ltd. www.go-faster.co.uk 2001-2009

PACKAGE DEFAULTS

Character VARCHAR2 definition         : N
Rebuild tables with redo logging      : N
Enable parallelism for table copy     : N
Enable parallel index build           : Y
Grant privileges to roles             : N
Name of update all role               :
Name of select all role               :
ID Prefix in files and project        : GFCBUILD
Drop indexes                          : Y
Pause commands in build script        : N
Explicitly specify schema             : Y
Block sample table statistics         : Y
Analyze table immediately after rebuild: N
Delete and lock statistics on GTTs    : Y
Force LONGs to CLOBs                  : N
Name of DDL trigger to disable on build:
Drop tables with PURGE option         : Y
Force rebuild if no change            : Y
Force descending index                : Y
```

NB: This procedure outputs via DBMS_OUTPUT. You will need to *set serveroutput on* in SQL*Plus to obtain any output.

HISTORY Procedure

This procedure lists the change history to the package.

Syntax

```
gfc_pspart.history;
```

There are no parameters to this procedure.

```
GFC_PSPART - Partitioned/Global Temporary Table DDL generator for PeopleSoft
(c)Go-Faster Consultancy Ltd. www.go-faster.co.uk 2001-2009
03.12.2002 - improved subrecord handling
11.02.2003 - correction to column sequencing on user
09.07.2003 - nologging facility added
10.07.2003 - oracle version detection added to control fix for 8.1.7.2 bug
05.09.2003 - corrected handling of peoplesoft long character columns
18.09.2003 - added trigger to prevent updates on tables whilst being rebuilt
09.10.2003 - tables and indexes set to logging enabled and parallel disabled,
parallel control variable added
28.10.2003 - supress partitioning for tables with long columns
29.10.2003 - pt version detection to enable new PT8.4 features
04.11.2003 - oracle 9i features, role name control
17.11.2003 - rename/drop index
07.01.2004 - explicit schema name, script control
22.03.2004 - oracle 9 varchars in characters
27.09.2004 - suppress disabled index build, but force disabled index to drop
18.04.2005 - support for PeopleSoft temporary tables
05.12.2006 - added handling for hash-partitioned only tables
13.12.2006 - remove partitioning column check
06.09.2007 - enhancement for PeopleTools 8.48, support partitioned function based indexes
08.11.2007 - support for range-list composite partitioning
14.02.2008 - selective list partition -v- range build, drop purge, all keys on subrecords
28.08.2008 - conversion to package procedure
16.09.2008 - DDL moved to gfcbuildtab
16.12.2008 - Single table build options can be combined
23.01.2009 - Partitioning columns in unique indexes must not be descending
01.04.2009 - Corrections to add partition scripts
18.04.2009 - Override Default Application Designer Project Name
03.06.2009 - Extended check descending partitioning columns in unique indexes to subrecords
23.04.2010 - Add ability to add range partitions by splitting
01.05.2010 - Correct setting new table LOGGING NOPARALLEL
05.05.2010 - Create table only if doesn't exist, add Colums when altering table
17.05.2010 - Temporary Record that are not declared to any AE can be GTT
08.06.2010 - Specified Source Table
16.06.2010 - Function Based Index support enhanced
01.10.2010 - Insert into copy table within PL/SQL block
02.10.2010 - Optionally reenable parallelism on existing table prior to copy
```

NB: This procedure outputs via DBMS_OUTPUT. You will need to *set serveroutput on* in SQL*Plus to obtain any output.

RESET_DEFAULTS Procedure

This procedure resets the variables in the system context back to their default values (see page 23).

Syntax

```
gfc_pspart.reset_defaults;
```

There are no parameters to this procedure.

SET_DEFAULTS Procedure

This procedure permits one or more variables in the system context to be set. This values are persistent.

Syntax

```
gfc_pspart.set_defaults(
(p_chardef         VARCHAR2 DEFAULT ''
,p_logging         VARCHAR2 DEFAULT ''
,p_parallel_table  VARCHAR2 DEFAULT ''
,p_parallel_index  VARCHAR2 DEFAULT ''
,p_force_para_dop  VARCHAR2 DEFAULT ''
,p_roles           VARCHAR2 DEFAULT ''
,p_scriptid        VARCHAR2 DEFAULT ''
,p_update_all      VARCHAR2 DEFAULT ''
,p_read_all        VARCHAR2 DEFAULT ''
,p_drop_index      VARCHAR2 DEFAULT ''
,p_pause           VARCHAR2 DEFAULT ''
,p_explicit_schema VARCHAR2 DEFAULT ''
,p_block_sample    VARCHAR2 DEFAULT ''
,p_build_stats     VARCHAR2 DEFAULT ''
,p_deletetempstats VARCHAR2 DEFAULT ''
,p_longtoclob      VARCHAR2 DEFAULT ''
,p_ddlenable       VARCHAR2 DEFAULT ''
,p_ddldisable      VARCHAR2 DEFAULT ''
,p_forcebuild      VARCHAR2 DEFAULT ''
,p_desc_index      VARCHAR2 DEFAULT ''
,p_repopdfltsub    VARCHAR2 DEFAULT ''
,p_repopnewmax     VARCHAR2 DEFAULT ''
,p_debug_level     INTEGER DEFAULT NULL
);
```

Parameters

Parameter	Description
p_chardef	Y = Use VARCHAR2 character length definition.
p_logging	N = DDL script includes NOLOGGING options when building tables and indexes
p_parallel_table	Y = set default parallelism on existing table prior to copy
p_parallel_index	Y = include parallel index build option
p_force_para_dop	Specify degree of parallelism in alter session commands in generated script to force parallelism in DML and DDL commands.
p_roles	Y= if roles are to be granted to tables
p_scriptid	ID string to be used in name of script and name of project generated.
p_update_all	Name of Role to be granted to permanent tables
p_read_all	Name Role to be granted to permanent tables
p_drop_index	Y = index dropped before rename, N = index is renamed to OLD_
p_pause	Y = Add pause commands to generated script
p_explicit_schema	Explicitly specify PeopleSoft owner ID in DDL script.
p_block_sample	Specify block sampling in update statistics script.
p_build_stats	Y = add gather statistics commands to build script.
p_deletetempstats	Y = delete and from Oracle 10g also lock statistics on temporary tables.
p_longtoclob	Y = always create long columns as CLOB, Otherwise only do so if PSOPTION.DATABASE_OPTIONS=2

p_ddlenable	Command to permit DDL to be issued on table – intended to disengage PSFT_DDL_LOCK trigger
	
	Eg.
	
	`gfc_pspart.set_defaults(p_ddlenable=>'BEGIN`
	`psft_ddl_lock.set_ddl_permitted(TRUE);END;' ||CHR(10)||'/');`
	Or
	
	`gfc_pspart.set_defaults(p_ddlenable=>'ALTER TRIGGER PSFT_DDL_LOCK DISABLE;');`

p_ddldisable	Command to prevent DDL from being issued on table – intended to reengage PSFT_DDL_LOCK trigger.

	Eg.
	`gfc_pspart.set_defaults(p_ddlenable=>'BEGIN psft_ddl_lock.set_ddl_permitted(FALSE);END;' ||CHR(10)||'/');`
	Or
	`gfc_pspart.set_defaults(p_ddlenable=>'ALTER TRIGGER PSFT_DDL_LOCK ENABLE;');`

p_drop_purge	Y = Add purge option to DROP TABLE commands on Oracle 10g or higher.
p_forcebuild	Debug option only – not supported
p_desc_index	Y include DESC keyword on columns specified in PeopleSoft as descending N = never use DESC keyword in CREATE INDEX commands.
p_repopdfltsub	N = exchange default list partition back into partitioned table when adding list sub-partitions Y = create new default list partition after adding new list sub-partitions and then copy the data back into the partitioned table from the exchange table.
p_repopnewmax	N = add new partitions by splitting MAXVALUE partition Y = exchange MAXVALUE partition out, add new partitions and rebuild and repopulate new MAXVALUE partition
p_debug_level	0 = off. Higher values add more debug information. Only to be used on advice from Go-Faster Consultancy Ltd.

TRUNCATE_TABLES Procedure

This procedure clears out the working storage and meta-data tables

Syntax

```
gfc_pspart.truncate_tables
(p_all BOOLEAN DEFAULT FALSE
);
```

Parameters

Parameter	Description
p_all	If true, all the working storage tables and meta-data tables are truncated. Otherwise, only the working storage tables are truncated. The meta-data tables only need to be truncated when different or new meta-data is being inserted.

SPOOLER Function

This function can be used to spool

Syntax

```
TYPE outrecset IS TABLE OF VARCHAR2(200);
FUNCTION spooler
(p_type NUMBER DEFAULT 0)
RETURN outrecset PIPELINED;
```

Parameters

Parameter	Description
p_type	0 = gfcbuild script. Similar to a PeopleSoft Alter by Recreation script 1 = gfcindex: Rebuild All indexes 2 = gfcstats: Refresh All statistics 3 = gfcalter: All missing partitions 4 = gfcarch1: Build archive tables (possibly in a different schema to the main PeopleSoft database). This script also includes exchange tables which are created when tables have either an archive policy, or a delete policy with a no-archive condition. 5 = gfcarch2: Grant privileges on PeopleSoft tables to archive schema

Example

```
rem gfcbuildspool.sql
rem (c) Go-Faster Consultancy Ltd.
rem 17.9.2008 - moved spool commands from gfcbuild.sql to this script

column line format a254
set head off feedback off echo off verify off pages 0 lines 1024 trimspool on
set termout off

column SPOOL_FILENAME   new_value SPOOL_FILENAME
select LOWER('gfcbuild_'||MAX(dbname)||'.sql') SPOOL_FILENAME from ps.psdbowner where
UPPER(ownerid) = user;
spool &&SPOOL_FILENAME
undefine SPOOL_FILENAME
select * from table(gfc_pspart.spooler(0));
spool off
…
```

MAIN Procedure

The main procedure generates all the DDL for all the tables specified. The three parameters specific tables or groups of tables, so that only a subset of the records specified in the meta-data can be built.

Syntax

```
gfc_pspart.main
(p_part_id     VARCHAR2 DEFAULT ''
,p_recname     VARCHAR2 DEFAULT ''
,p_rectype     VARCHAR2 DEFAULT 'A'
,p_projectname VARCHAR2 DEFAULT ''
);
```

Parameters

Parameter	Description
p_part_id[51]	Name of partition ID. A simple pattern string may be specified as would be used with the LIKE operator. Partitioned records whose PART_IDs match this parameter will be generated.
p_recname	Name of record to be generated. A simple pattern string may be specified as would be used with the LIKE operator.
p_rectype	Restricts scope of package to records in particular groups. P = Partitioned Tables including any corresponding archive and exchange tables and relavent privileges T = Global Temporary Tables only A = All tables, both Partitioned and Global Temporary. R = Archive and Exchange tables only for partitioned tables and relavent privileges Default value is A
p_projectname	The name of an Application Desginer project can be specified, and then only the records in the project will be built. Use this option when an upgrade project is being applied and you want to build just the tables in the project.

[51] Added 11th February 2013