# Technical Manual

## Managing Oracle Table Partitioning in PeopleSoft Applications with GFC\_PSPART package

Prepared By David Kurtz, Go-Faster Consultancy Ltd.

Technical Manual

Version 1.09 – <http://www.go-faster.co.uk/gfc_pspart.manual.pdf>

Monday 2 March 2015

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

File: gfc\_pspart.manual.v1.09.docx, 2 March 2015

Contents

[Introduction 3](#_Toc413080915)

[Delivered Files 4](#_Toc413080916)

[Installation 5](#_Toc413080917)

[Operation 6](#_Toc413080918)

[Example Output 8](#_Toc413080919)

[Meta-Data 11](#_Toc413080920)

[Meta-data Tables 11](#_Toc413080921)

[GFC\_PART\_INDEXES 11](#_Toc413080922)

[GFC\_PART\_LISTS 13](#_Toc413080923)

[GFC\_PART\_TABLES 14](#_Toc413080924)

[GFC\_PART\_RANGES 17](#_Toc413080925)

[GFC\_PART\_RANGE\_LISTS 18](#_Toc413080926)

[GFC\_PS\_IDXDDLPARM 18](#_Toc413080927)

[GFC\_PS\_INDEXDEFN 19](#_Toc413080928)

[GFC\_PS\_KEYDEFN 19](#_Toc413080929)

[GFC\_TEMP\_TABLES 20](#_Toc413080930)

[Working Storage Tables 21](#_Toc413080931)

[GFC\_PS\_TABLES 21](#_Toc413080932)

[GFC\_PS\_TAB\_COLUMNS 22](#_Toc413080933)

[GFC\_ORA\_TAB\_COLUMNS 23](#_Toc413080934)

[GFC\_DDL\_SCRIPT 23](#_Toc413080935)

[Context 24](#_Toc413080936)

[Example Meta-Data 26](#_Toc413080937)

[Example General Ledger Meta-Data 26](#_Toc413080938)

[Example Global Payroll Meta-Data 28](#_Toc413080939)

[Global Indexes 34](#_Toc413080940)

[Global Payroll Streams 35](#_Toc413080941)

[Function Based Indexes 36](#_Toc413080942)

[List Partitioning 38](#_Toc413080943)

[Selectively Building Sub-Partitions 40](#_Toc413080944)

[Setting Context Values 41](#_Toc413080945)

[Globally Partitioned Indexes 41](#_Toc413080946)

[Example Time & Labor Meta-Data 42](#_Toc413080947)

[Archiving Time-Based Range Partitioning 42](#_Toc413080948)

[Meta-Data Report 44](#_Toc413080949)

[Partitioned Tables 44](#_Toc413080950)

[Range Partitioning 44](#_Toc413080951)

[List Partitioning 45](#_Toc413080952)

[Global and Globally Partitioned Indexes 45](#_Toc413080953)

[Specified Tablespaces 46](#_Toc413080954)

[Package GFC\_PSPART 47](#_Toc413080955)

[BANNER Procedure 47](#_Toc413080956)

[DISPLAY\_DEFAULTS Procedure 48](#_Toc413080957)

[HISTORY Procedure 49](#_Toc413080958)

[RESET\_DEFAULTS Procedure 50](#_Toc413080959)

[SET\_DEFAULTS Procedure 50](#_Toc413080960)

[TRUNCATE\_TABLES Procedure 52](#_Toc413080961)

[SPOOLER Function 53](#_Toc413080962)

[MAIN Procedure 54](#_Toc413080963)

### 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]](#footnote-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.

### 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 26).
* *gp-partdata.sql*: This utility was originally developed for use with PeopleSoft Global Payroll Systems were Payroll ‘Streaming[[2]](#footnote-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 6).
* *gfcbuildone.sql*: This is a variant of *gfcbuild.sql* which builds the DDL scripts for named tables only (see Operation on page 6).
* *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.

### Installation

The package is designed to be installed and run in the owner of the PeopleSoft database – by default SYSADM[[3]](#footnote-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.

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

### 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]](#footnote-4)

@@partdata[[5]](#footnote-5)

--execute gfc\_pspart.truncate\_tables;[[6]](#footnote-6)

--all tables

execute gfc\_pspart.main;[[7]](#footnote-7)

--just generate global temporary tables

--execute gfc\_pspart.main(p\_rectype => 'T');[[8]](#footnote-8)

--just generate named tables

--execute gfc\_pspart.main(p\_recname => 'GP\_PI\_GEN\_SOVR', p\_rectype => 'P');[[9]](#footnote-9)

--extract script to file

@@gfcbuildspool.sql[[10]](#footnote-10)

set head on feedback on termout on pages 50

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]](#footnote-11)

--set termout off

--@@partdata

--set termout on

execute gfc\_pspart.truncate\_tables;[[12]](#footnote-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]](#footnote-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]](#footnote-14)

--pause

--extract script to file

@@gfcbuildspool.sql

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]](#footnote-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]](#footnote-16)

…

WHENEVER SQLERROR CONTINUE[[17]](#footnote-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

)

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]](#footnote-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]](#footnote-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

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]](#footnote-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

/

# Appendix

### 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\_TYPE | VARCHAR2(1) | 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 |

##### 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]](#footnote-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. |

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\_RANGE\_LISTS

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. |
| 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]](#footnote-22). |

##### 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 53)  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]](#footnote-23) | NULL | Command to permit DDL to be issued against table |
| 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]](#footnote-24), 'R', 'LEDGER', 0, 'GLLARGE', 'PSINDEX',

'PCTUSED 90 PCTFREE \*\*PCTFREE\*\*'[[25]](#footnote-25), 'PCTFREE \*\*PCTFREE\*\*', 'Y', NULL, 'FOR ALL COLUMNS SIZE 1', NULL);

INSERT INTO gfc\_part\_ranges[[26]](#footnote-26) VALUES('GL',2000,' 2000', '2001,0', NULL[[27]](#footnote-27), NULL, 'PCTFREE 0', 'PCTFREE 0'[[28]](#footnote-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');

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]](#footnote-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]](#footnote-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');

##### 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]](#footnote-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\*/

'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]](#footnote-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\*/

))

/

-----------------------------------------------------------------------------------------------------------

--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]](#footnote-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

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]](#footnote-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

/

###### 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]](#footnote-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]](#footnote-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);

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]](#footnote-37)

(part\_id, part\_no, part\_name, list\_value)

VALUES ('GP',9999,'Z\_OTHERS','DEFAULT')

/

-----------------------------------------------------------------------------------------------------------

--monthly partitions for Pensioners[[38]](#footnote-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]](#footnote-39)

FROM dba\_objects

WHERE rownum <= 3

) y

ORDER BY 1,2,3

/

-----------------------------------------------------------------------------------------------------------

--lunar monthly partitions for lunar and weekly[[40]](#footnote-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]](#footnote-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

/

###### 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]](#footnote-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]](#footnote-43)

OR (list\_name like 'UK%' AND NOT range\_name IN('02','03','04','05','06','07','08')))[[44]](#footnote-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'

--;

###### 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 50).

-----------------------------------------------------------------------------------------------------------

--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]](#footnote-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')

/

##### Example Time & Labor Meta-Data

###### Archiving Time-Based Range Partitioning

UPDATE gfc\_part\_tables

SET arch\_flag = 'A'[[46]](#footnote-46)

, arch\_schema = 'PSARCH'[[47]](#footnote-47)

WHERE part\_id IN('SCH')

OR recname IN('AUDIT\_SCH\_TBL')

/

UPDATE gfc\_part\_tables

SET noarch\_condition = 'sch\_adhoc\_ind = ''1'''[[48]](#footnote-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]](#footnote-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

) 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]](#footnote-50)

###### 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](http://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 24).

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 21).

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\_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\_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]](#footnote-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. |

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

   Partition elimination to reduce scans of tables and indexes.

   Separation of data to reduce consistent read during parallel batch processing.. [↑](#footnote-ref-1)
2. PeopleSoft coined this term to describe running certain payroll processes in several concurrently executing processes. [↑](#footnote-ref-2)
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. [↑](#footnote-ref-3)
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. [↑](#footnote-ref-4)
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. [↑](#footnote-ref-5)
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. [↑](#footnote-ref-6)
7. Running the main procedure with no paramters builds the DDL for ALL tables specified in the meta-data. [↑](#footnote-ref-7)
8. It is possible to build the DDL just some tables, this commented out command would build [↑](#footnote-ref-8)
9. This command generates the DDL for a specified partitioned table. [↑](#footnote-ref-9)
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. [↑](#footnote-ref-10)
11. In this case the meta-data is not being truncated and reloaded, because it has not changed. [↑](#footnote-ref-11)
12. Instead, we will only truncate the working storage tables. [↑](#footnote-ref-12)
13. In this case all tables whose names begin with GP\_RSLT. This is an alternative to explicitly listing them. [↑](#footnote-ref-13)
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). [↑](#footnote-ref-14)
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. [↑](#footnote-ref-15)
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. [↑](#footnote-ref-16)
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. [↑](#footnote-ref-17)
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. [↑](#footnote-ref-18)
19. The source table is explicitly locked in case any transactions were in flight when the trigger was created. [↑](#footnote-ref-19)
20. The newly created objects replace the old ones through a series or rename operations, and the old objects are dropped. [↑](#footnote-ref-20)
21. This parameter can be used when you want to archive data and also reorganise the partitioned tables. [↑](#footnote-ref-21)
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. [↑](#footnote-ref-22)
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. [↑](#footnote-ref-23)
24. The table will be range partitioned on the combination of FISCAL\_YEAR and ACCOUNTING\_PERIOD. Two columns, but partitioned in a single dimension. [↑](#footnote-ref-24)
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. [↑](#footnote-ref-25)
26. A row is created in GFC\_PART\_RANGES for each range partition. [↑](#footnote-ref-26)
27. Tablespaces have not been specified at range partition level, so all partitions will go into the tablespaces specified at TABLE and INDEX level. [↑](#footnote-ref-27)
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. [↑](#footnote-ref-28)
29. Additional index data is added to gfc\_ps\_idxddlparm. [↑](#footnote-ref-29)
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 onit, but it still leads on FISCAL\_YEAR and ACCOUNTING\_PERIOD, and so is also locally partitioned [↑](#footnote-ref-30)
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. [↑](#footnote-ref-31)
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. [↑](#footnote-ref-32)
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. [↑](#footnote-ref-33)
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. [↑](#footnote-ref-34)
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. [↑](#footnote-ref-35)
36. Index PSZTL\_PAYABLE\_TIME\_SPARE will also be created on table PS\_TL\_PAYABLE\_TIME. This index will be locally partitioned index. [↑](#footnote-ref-36)
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. [↑](#footnote-ref-37)
38. One list partition per year. Each partition contains 12 calendar group IDs. [↑](#footnote-ref-38)
39. Partitions will be created for 3 tax years starting with 2008-09. [↑](#footnote-ref-39)
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 [↑](#footnote-ref-40)
41. The 53rd weekly period will be added to the 13th partition for that year. [↑](#footnote-ref-41)
42. For ease of processing a table with ever combination of range and list is built, and then unwanted partitions will be marked. [↑](#footnote-ref-42)
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. [↑](#footnote-ref-43)
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. [↑](#footnote-ref-44)
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)||'/'); [↑](#footnote-ref-45)
46. Tables to be archived or purged by dropping partitions can be identified in the meta data. [↑](#footnote-ref-46)
47. In this case the data will be moved to an archive table with the same name but in a different schema. [↑](#footnote-ref-47)
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. [↑](#footnote-ref-48)
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. [↑](#footnote-ref-49)
50. Note that globally partitioned indexes must have an unlimited upper range, so the last value must use a MAXVALUE operator. [↑](#footnote-ref-50)
51. Added 11th February 2013 [↑](#footnote-ref-51)