

311-CD-106-005

EOSDIS Core System Project

**Release 4 Planning and Data Processing
Subsystem Database Design and
Schema Specifications
for the ECS Project**

Final

March 1999

**This document has not yet been approved by the
Government for general use or distribution.**

Raytheon Systems Company
Upper Marlboro, Maryland

Release 4 Planning and Data Processing Subsystem Database Design and Schema Specifications for the ECS Project

Final

March 1999

Prepared Under Contract NAS5-60000
CDRL Item #050

RESPONSIBLE ENGINEER

Maureen Muganda /s/ 2/26/99
Maureen Muganda Date
EOSDIS Core System Project

SUBMITTED BY

Mary Armstrong /s/ 2/26/99
Mary Armstrong, Development Engineering Manager Date
EOSDIS Core System Project

Raytheon Systems Company
Upper Marlboro, Maryland

This page intentionally left blank.

Preface

This document describes the data design and database specification for the Planning and Data Processing Subsystem. It is one of eight documents comprising the detailed database design specifications for each of the ECS subsystems.

The subsystem database design specifications for the as delivered system include:

- 311-CD-102 Data Management (DM) Subsystem Database Design and Database Schema Specifications for the ECS Project
- 311-CD-103 Ingest Subsystem Database Design and Database Schema Specifications for the ECS Project
- 311-CD-104 Interoperability Subsystem (IOS) Database Design and Database Schema Specifications for the ECS Project
- 311-CD-105 Management Support Subsystem (MSS) Database Design and Database Schema Specifications for the ECS Project
- 311-CD-106 Planning and Data Processing Subsystem (PDPS) Database Design and Database Schema Specifications for the ECS Project
- 311-CD-107 Science Data Server (SDSRV) Subsystem Database Design and Database Schema Specifications for the ECS Project
- 311-CD-108 Storage Management (STMGMT) Subsystem Database Design and Database Schema Specifications for the ECS Project
- 311-CD-109 Subscription Server (SUBSRV) Subsystem Database Design and Database Schema Specifications for the ECS Project

This submittal meets the milestone specified in the Contract Data Requirements List (CDRL) of NASA Contract NAS5-60000. It is a formal contract deliverable with an approval code 1. It requires Government review and approval prior to acceptance and use. This document is under ECS contractor configuration control. Once approved, contractor approved changes will be handled in accordance with Class I and Class II change control requirements described in the EOS Configuration Management Plan, and changes to this document shall be made by Document Change Notice (DCN) or by complete revision.

Entity Relationship Diagrams (ERDs) presented in this document have been exported directly from tools and some cases contain too much detail to be easily readable within hard copy page constraints. The reader is encouraged to view these drawings on-line using the Portable Document Format (PDF) electronic copy available via the ECS Data Handling System (ECS) on the world-wide web at <http://edhs1.gsfc.nasa.gov>.

Any questions should be addressed to:

Data Management Office
The ECS Project Office
Raytheon Systems Company
1616 McCormick Drive
Upper Marlboro, MD 20774-5301

Abstract

This document outlines Release 4 Drop 4PX “as-built” database design and database schema of the Planning and Data Processing Subsystem database, including the physical layout of the database and initial installation parameters.

Keywords: data, database, design, configuration, database installation, scripts, security, data model, data dictionary, replication, performance tuning, SQL server, database security, replication, database scripts

This page intentionally left blank.

Change Information Page

List of Effective Pages			
Page Number	Issue		
Title	Submitted as Final		
iii through xii	Submitted as Final		
1-1 and 1-2	Submitted as Final		
2-1 and 2-2	Submitted as Final		
3-1 and 3-2	Submitted as Final		
4-1 through 4-196	Submitted as Final		
5-1 through 5-4	Submitted as Final		
6-1 through 6-4	Submitted as Final		
7-1 and 7-2	Submitted as Final		
8-1 and 8-2	Submitted as Final		
A-1 through A-14	Submitted as Final		
AB-1 through AB-8	Submitted as Final		
Document History			
Document Number	Status/Issue	Publication Date	CCR Number
311-CD-106-001	Draft	January 1998	97-1755
311-CD-106-002	Draft	May 1998	98-0620
311-CD-106-003	Draft	July 1998	98-0866
311-CD-106-004	Draft	December 1998	98-1267
311-CD-106-005	Submitted as Final	March 1999	99-0166

This page intentionally left blank.

Contents

Preface

Abstract

1. Introduction

1.1	Identification	1-1
1.2	Scope	1-1
1.3	Purpose	1-1
1.4	Audience	1-1

2. Related Documents

2.1	Applicable Documents	2-1
2.2	Information Documents	2-2

3. Database Configurations

3.1	Server Configurations	3-1
3.2	Storage Device Layouts	3-1

4. Data Design

4.1	Database Overview	4-1
4.1.1	Physical Data Model Entity Relationship Diagram	4-1
4.1.2	Tables	4-2
4.1.3	Columns	4-29
4.1.4	Column Domains	4-48
4.1.5	Rules	4-48

4.1.6	Defaults	4-48
4.1.7	Views	4-49
4.1.8	Integrity Constraints	4-50
4.1.9	Triggers	4-55
4.1.10	Stored Procedures	4-184
4.2	Flat File Usage	4-193
4.2.1	File Descriptions	4-193
4.2.2	Field Specifications	4-193

5. Performance and Tuning Factors

5.1	Indexes	5-1
5.2	Segments	5-4
5.3	Named Caches	5-4

6. Database Security

6.1	Initial Users	6-1
6.2	Groups	6-2
6.3	Roles	6-2
6.4	Object Permissions	6-3

7. Scripts

7.1	Installation Scripts	7-1
7.2	Backup/Recovery Scripts	7-1
7.3	Miscellaneous Scripts	7-1

List of Figures

4-1. ERD Key	4-1
--------------------	-----

List of Tables

4-6. Flat File Descriptions	4-193
4-7. Flat File Field Specifications	4-193

Appendix A- PDPS Database Schema Diagrams

Abbreviations and Acronyms

This page intentionally left blank.

1. Introduction

1.1 Identification

This Planning and Data Processing Subsystem (PDPS) Database Design and Database Schema Specification document, Contract Data Requirement List (CDRL) Item #050, whose requirements are specified in Data Item Description (DID) 311/DV1, is a required deliverable under the Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS), Contract NAS5-60000.

1.2 Scope

The PDPS Database Design and Database Schema Specification document describes the data design and database specifications to support the data requirements of Release 4 Drop 4PX PDPS software.

1.3 Purpose

The purpose of the PDPS Database Design and Database Schema Specification document is to support the maintenance of PDPS data and databases throughout the life cycle of ECS. This document communicates the database implementation in sufficient detail to support ongoing configuration management.

1.4 Audience

This document is intended to be used by ECS maintenance and operations staff. The document is organized as follows:

Section 1 provides information regarding the identification, purpose, scope and audience of this document.

Section 2 provides a listing of the related documents, which were used as a source of information for this document.

Section 3 provides a mapping of databases to hardware components.

Section 4 contains the PDPS physical data model, which is the database tables, triggers, stored procedures, and flat files.

Section 5 provides a description of database performance and tuning features such as indexes, caches, and data segments.

Section 6 provides a description of the security infrastructure used and list of the users, groups, and permissions available upon initial installation.

Section 7 provides a description of database and database related scripts used for installation and backup/recovery.

2. Related Documents

2.1 Applicable Documents

The following documents, including Internet links, are referenced in this document, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this volume:

305-CD-100	Release 4 Segment Design Specification for the ECS Project
920-TDG-009	DAAC Hardware Database Mapping/GSFC
920-TDN- 009	DAAC Hardware Database Mapping/NSIDC
920-TDE-009	DAAC Hardware Database Mapping/EDC
920-TDL-009	DAAC Hardware Database Mapping/LARC
920-TDS-009	DAAC Hardware Database Mapping/SMC
920-TDG-010	DAAC Database Configuration/GSFC
920-TDN-010	DAAC Database Configuration/NSIDC
920-TDE-010	DAAC Database Configuration/EDC
920-TDL-010	DAAC Database Configuration/LARC
920-TDS-010	DAAC Database Configuration/SMC
922-TDG-013	Disk Partitions/GSFC
922-TDN-013	Disk Partitions/NSIDC
922-TDE-013	Disk Partitions/EDC
922-TDL-013	Disk Partitions/LARC
922-TDS-013	Disk Partitions/SMC

These documents are maintained as part of the ECS baseline and available on the World Wide Web at the URL: <http://cmdm.east.hitc.com/baseline>. Please note that this is a partial mirror site in that some items are not available (they are identified) since this is OPEN to all. This site may also be reached through the EDHS homepage. Scroll page to the connections line and click on the ECS Baseline Information System link.

2.2 Information Documents

The following documents, although not directly applicable, amplify or clarify the information presented in this document. These documents are not binding on this document.

313-CD-006 Release 4 CSMS/SDPS Internal ICD for the ECS Project

609-CD-003 Release 4 Operations Tools Manual for the ECS Project

611-CD-004 Release 4 Mission Operation Procedures for the ECS Project

These documents are accessible via the EDHS homepage.

3. Database Configurations

3.1 Server Configurations

The database configuration of the server varies from DAAC to DAAC based on individualized DAAC requirements and hardware availability. These DAAC-specific database configurations are detailed on the following documents:

920-TDG-009	DAAC Hardware Database Mapping/GSFC
920-TDN- 009	DAAC Hardware Database Mapping/NSIDC
920-TDE-009	DAAC Hardware Database Mapping/EDC
920-TDL-009	DAAC Hardware Database Mapping/LARC
920-TDS-009	DAAC Hardware Database Mapping/SMC

These documents are maintained as part of the ECS baseline and available on the world-wide web at the URL: <http://cmdm.east.hitc.com/baseline>. Please note that this is a partial mirror site in that some items are not available (they are identified) since this is OPEN to all. This site may also be reached through the EDHS homepage. Scroll page to the connections line and click on the ECS Baseline Information System link.

3.2 Storage Device Layouts

Storage Device layouts, disk partitions, vary from DAAC to DAAC based on the amount of data storage expected to be needed to accommodate a particular DAAC's storage requirements. Disk partitions for the PDPS server at each DAAC are detailed in the following documents:

922-TDG-013	Disk Partitions/GSFC
922-TDN-013	Disk Partitions/NSIDC
922-TDE-013	Disk Partitions/EDC
922-TDL-013	Disk Partitions/LARC
922-TDS-013	Disk Partitions/SMC

These documents are maintained as part of the ECS baseline and available on the world-wide web at the URL: <http://cmdm.east.hitc.com/baseline>. Please note that this is a partial mirror site in that some items are not available (they are identified) since this is OPEN to all. This site may also be reached through the EDHS homepage. Scroll page to the connections line and click on the ECS Baseline Information System link.

This page intentionally left blank.

4. Data Design

4.1 Database Overview

The PDPS database implements the large majority of the persistent data requirements for the PDPS subsystem. The database is designed in such a manner as to satisfy business policy while maintaining data integrity and consistency. Database tables are implemented using the Sybase Relational Database Management system (DBMS). All components of the PDPS database are described in the sections which follow, in sufficient detail to support maintenance needs.

4.1.1 Physical Data Model Entity Relationship Diagram

The Entity Relationship Diagram(ERD) presents a schematic depiction of the PDPS physical data model. The ERDs presented here for the PDPS database were produced using the S-Designer Data Architect Computer Aided Software Engineering (CASE) tool. ERDs represent the relationship between entities or database tables. The key for the symbols used in the ERDs follows.

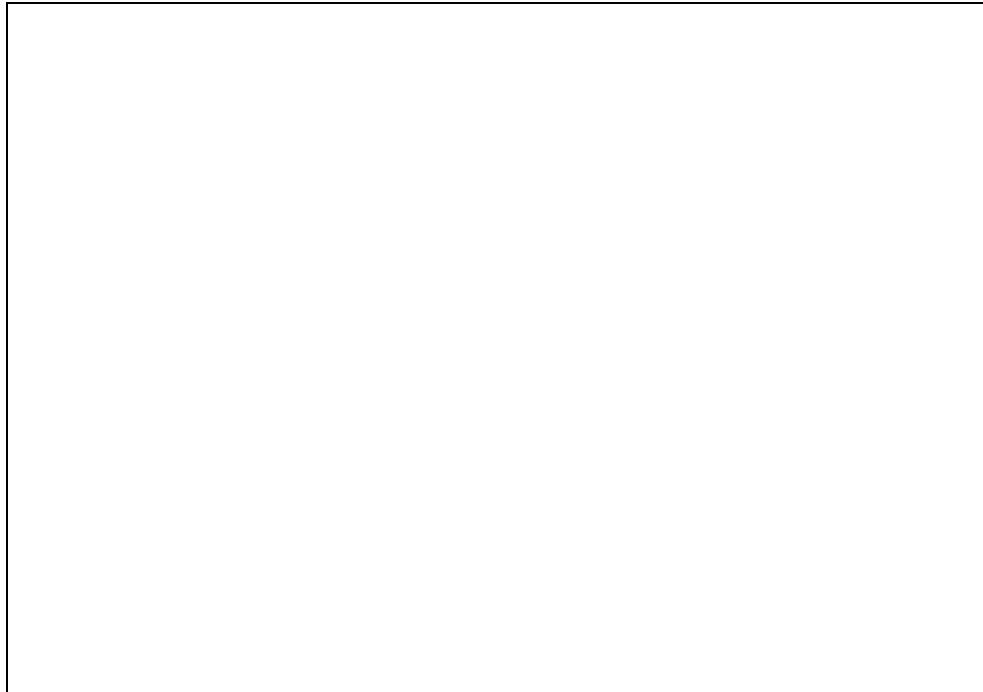


Figure 4-1. ERD Key

The ERDs for the PDPS database are shown in Appendix A – Planning and Data Processing Subsystem ERDs.

4.1.2 Tables

A listing of each the tables in the PDPS database is given here. A brief definition of each of these tables follows.

Table	Table
DpPrAutosysMapList	PIPathModel
DpPrCreationQueue	PIPgeDetailParameters
DpPrDeletableGranules	PIPgeDetailTile
DpPrDiskAllocation	PIPgeDetailTime
DpPrDprDependList	PIPgeMaster
DpPrExecutable	PIPgeOrbitModel
DpPrFile	PIPgePerformance
DpPrGranuleLocation	PIPgePriority
DpPrPcf	PIPPlans
DpPrPge	PIPrMetadataValue
DpPrPgeExitDependency	PIProductionRequest
DpPrPgeExitMessage	PIProductionStrategy
DpPrResourceLock	PIPrParameter
DpPrRpcID	PIPRPriority
DpPrRscCpuRamAllocation	PIPrVsPlan
PIAlternate	PIRescUseThresh
PIAlternateInputValues	PIResource
PIAssociatedScienceData	PIResourceRequirement
PIDataGranuleShort	PIRoutineArrival
PIDataProcessingRequest	PIRscComputer
PIDataSourceMaster	PIRscDiskPartition
PIDataTypeFiles	PIRscRealComputer
PIDataTypeMaster	PIRscString
PIDataTypeReq	PITileCoordinates
PIDprCompletion	PITileSchemaShort
PIDprData	PITimer
PIEsdtParam	PIUsedByCenter
PIEsdtParmValues	PIUserPriority
PIFileTypeReq	RpActivityType
PIGroundEvent	RpReservationInterval
PIGroundEventResourceExplosion	RpResourceReservation
PIMetadataChecks	RpRscPlanComments
PINotification	RpRsvRscExplosion
PIOOutputDataYield	PIOOutputFileType

Table: DpPrAutosysMapList

Description

Maintains a mapping of instances of Autosys to physical resources.

Column List

Name	Code	Type	Primary Key	Mandatory
autosysid	autosysId	varchar(20)	No	Yes
autosysidkey	autosysIdKey	int	Yes	Yes
resourcestring	resourceString	varchar(20)	Yes	Yes

Table: DpPrCreationQueue

Description

Maintains a list of DPRs currently being processed by Autosys.

Column List

Name	Code	Type	Primary Key	Mandatory
autosysid	autosysId	varchar(20)	No	Yes
dprid	dprId	varchar(27)	Yes	Yes
hold	hold	bit	No	Yes
priority	priority	int	No	Yes

Table: DpPrDiskAllocation

Description

Describes the disk resources allocated for a job.

Column List

Name	Code	Type	Primary Key	Mandatory
computerid	computerId	int	No	Yes
diskallocationactual	diskAllocationActual	float	No	No
diskallocationid	diskAllocationId	numeric(9,0)	Yes	Yes
path	path	varchar(255)	No	No
diskallocationsize	diskAllocationSize	float	No	No
diskallocationtype	diskAllocationType	smallint	No	Yes
diskallocationuser	diskAllocationUser	varchar(27)	No	No
diskpartitionid	diskPartitionId	int	No	Yes

Table: DpPrDeletableGranules

Description

Identifies a set of data granule files that could possibly be deleted in order to reclaim disk storage.

Column List

Name	Code	Type	Primary Key	Mandatory
sequence	sequence	Numeric (9,0)	Yes	Yes
granuleId	granuleId	Varchar(130)	No	Yes
granuleSize	granuleSize	float	No	Yes
machineId	machineId	Varchar(60)	No	Yes
partitionSize	PartitionSize	float	No	Yes
diskUsage	diskUsage	float	No	Yes

Table: DpPrDprDependList

Description

A list of DPRs which depend on the completion of the current DPR.

Column List

Name	Code	Type	Primary Key	Mandatory
dependentdprid	dependentDprId	varchar(27)	Yes	Yes
dprid	dprId	varchar(27)	Yes	Yes

Table: DpPrExecutable

Description

Contains information about a particular execution of a PGE.

Column List

Name	Code	Type	Primary Key	Mandatory
exec_location	execLocation	varchar(240)	No	No
exec_name	execName	varchar(60)	Yes	Yes
exec_permission	execPermission	char(5)	No	Yes
exec_shell	execShell	varchar(10)	No	Yes
execlayer	execLayer	int	No	No
execmachine	execMachine	varchar(60)	Yes	Yes
operatingsystem	operatingSystem	varchar(60)	No	No
sswid	sswid	varchar(17)	Yes	Yes

Table: DpPrFile

Description

Contains information about files housing data granules.

Column List

Name	Code	Type	Primary Key	Mandatory
datainfofilesize	fileSize	float	No	No
datainfopath	path	varchar(255)	No	No
filename	fileName	varchar(140)	No	Yes
filenumber	fileNumber	int	No	No
filetype	fileType	int	No	No
machineid	machineId	varchar(60)	No	Yes
numberofusage	numberOfUsage	int	No	No
sequencenumber	sequenceNumber	int	No	No
universalreference	universalReference	varchar(255)	No	Yes
fileld	fileld	numeric(9)	Yes	No

Table: DpPrGranuleLocation

Description

Contains information about the physical location of a data granule.

Column List

Name	Code	Type	Primary Key	Mandatory
dprId	dprId	varchar(27)	No	No
granuleId	granuleId	varchar(130)	Yes	Yes
machineid	machineId	varchar(60)	Yes	Yes
stagestate	stageState	int	No	Yes

Table: DpPrPcf

Description

Information about a Process Control File (PCF).

Column List

Name	Code	Type	Primary Key	Mandatory
dpr_id	dprId	varchar(29)	Yes	Yes
pcf_location	pcfLocation	varchar(240)	No	No
pcf_name	pcfName	varchar(60)	No	Yes
pcf_permission	pcfPermission	char(5)	No	Yes
pcf_type	pcfType	int	Yes	Yes

Table: DpPrPge

Description

Information describing a PGE.

Column List

Name	Code	Type	Primary Key	Mandatory
cpu_resource	pgeCpu	int	No	No
dprid	dprId	varchar(29)	Yes	Yes
execmachine	execMachine	varchar(60)	No	No
execname	execName	varchar(60)	No	No
pge_commands	pgeCommands	varchar(120)	No	Yes
pge_environment	pgeEnvironment	varchar(240)	No	No
pge_shell	pgeShell	varchar(30)	No	Yes
pge_state	pgeState	int	No	Yes
sswid	sswid	varchar(17)	No	No

Table: DpPrPgeExitDependency

Description

Information about PGE dependencies upon termination.

Column List

Name	Code	Type	Primary Key	Mandatory
dependencysswid	dependencySswId	varchar(17)	Yes	Yes
exitcode	exitCode	int	Yes	Yes
exitoperation	exitOperation	varchar(3)	No	No
pgesswid	pgeSswId	varchar(17)	Yes	Yes

Table: DpPrPgeExitMessage

Description

Messages associated with the termination of a PGE.

Column List

Name	Code	Type	Primary Key	Mandatory
exitcode	exitCode	int	Yes	Yes
exitmessage	exitMessage	varchar(240)	No	No
pgesswid	pgeSswId	varchar(17)	Yes	Yes

Table: DpPrResourceLock

Description

Information for managing resource locking during data processing.

Column List

Name	Code	Type	Primary Key	Mandatory
attempts	attempts	int	No	No
ecsunit	ecsUnit	int	No	No
jobid	jobId	varchar(40)	Yes	Yes
priority	priority	int	No	No
state	state	int	No	No

Table: DpPrRpcID

Description

Processing table used in fault recovery.

Column List

Name	Code	Type	Primary Key	Mandatory
object	object	varchar(200)	No	Yes
readableTag	readableTag	varchar(40)	Yes	Yes

Table: DpPrRscCpuRamAllocation

Description

Information about memory allocation for data processing.

Column List

Name	Code	Type	Primary Key	Mandatory
computerid	computerId	int	Yes	Yes
cpuramallocationcpucount	cpuRamAllocationCpuCount	int	No	No
cpuramallocationjobid	cpuRamAllocationJobId	varchar(40)	Yes	Yes
cpuramallocationramsize	cpuRamAllocationRamSize	float	No	No
allocState	allocState	int	No	No

Table: PIAlternate

Description

Contains information about alternate inputs to PGEs.

Column List

Name	Code	Type	Primary Key	Mandatory
data_type_id	dataTypeld	varchar(20)	Yes	Yes
default_order	defaultOrder	int	No	No
default_timer	defaultTimer	integer	No	No
logicalId	logicalId	int	Yes	Yes
pge_id	pgelid	varchar(17)	Yes	Yes
primary_type	primaryType	varchar(20)	No	No
runtimeid	runTimeld	int	No	No
temp_flag	temporalFlag	bit	No	Yes
waitfor	waitForFlag	bit	No	Yes

Table: PIAlternateInputValues

Description

Information about alternate inputs to a production request.

Column List

Name	Code	Type	Primary Key	Mandatory
datatype	dataType	varchar(20)	Yes	Yes
logicalid	logicalId	int	Yes	Yes
order	theOrder	int	No	No
primarytype	primaryType	varchar(20)	No	No
productionrequestid	productionRequestId	varchar(20)	Yes	Yes
temporalflag	temporalFlag	bit	No	Yes
timerwait	timerWait	int	No	No
type	type	int	No	No

Table: PIAssociatedScienceData

Description

Associates a list of science data with an output granule.

Column List

Name	Code	Type	Primary Key	Mandatory
pgeld	pgeld	varchar(17)	Yes	Yes
datatype	dataType	varchar(20)	Yes	Yes
logicalId	logicalId	int	Yes	Yes

Table: PlAuxilliaryLogicalIds

Description

Contains lists of auxiliary logical IDs associated with input granules.

Column List

Name	Code	Type	Primary Key	Mandatory
pgeld	pgeld	varchar(17)	Yes	Yes
primarylogicalid	PrimarylogicalId	int	Yes	Yes
auxiliarylogicalid	auxiliaryLogicalId	int	Yes	Yes
auxiliarySequence-Number	auxiliarySequence-Number	int	No	Yes

Table: PlDataGranuleShort

Description

Basic information describing a data granule.

Column List

Name	Code	Type	Primary Key	Mandatory
absolutedeletetime	absoluteDeletionTime	datetime	No	No
availability	availability	bit	No	Yes
availability_actual	availabilityActual	datetime	No	No
availability_predicted	availabilityPredicted	datetime	No	No
baseline_time	baselineTime	datetime	No	No
data_type_id	dataTypeId	varchar(20)	No	Yes
granule_id	granuleId	varchar(130)	Yes	Yes
granalesize	granuleSize	float	No	No
predicted_staging_time	predictedStagingTime	integer	No	No
sciencegroup	scienceGroup	varchar(5)	No	No
stagestate	stageState	int	No	No
start_time	startTime	datetime	No	No
stop_time	stopTime	datetime	No	No
tile_id	tileId	int	No	No
UR	universalReference	varchar(255)	No	No
version	version	int	No	No

Name	Code	Type	Primary Key	Mandatory
timeStamp	timeStamp	datetime	No	No

Table: PIDataProcessingRequest

Description

Information describing a Data ProcessingRequest (DPR).

Column List

Name	Code	Type	Primary Key	Mandatory
actual_start	actualStartTime	datetime	No	No
alarm_time	alarmTime	datetime	No	No
autosysid	autoSysId	varchar(20)	No	No
baseline_time	baselineTime	datetime	No	No
completion_state	completionState	varchar(10)	No	No
data_start_time	dataStartTime	datetime	No	No
data_stop_time	dataStopTime	datetime	No	No
dpr_collection_id	dprCollectionId	varchar(10)	No	No
dpr_id	dprId	varchar(29)	Yes	Yes
latemessagesent	lateMessagesSent	int	No	No
name	name	varchar(27)	No	No
pge_id	pgeId	varchar(17)	No	Yes
predicted_start	predictedStart	datetime	No	No
priority	priority	int	No	No
request_id	productionRequestId	varchar(20)	No	Yes
runstate	runState	varchar(20)	No	No
sswid	sswid	varchar(17)	No	Yes
timeStamp	timeStamp	datetime	No	No

Table: PlDataSourceMaster

Description

Information about when external data will arrive in ECS.

Column List

Name	Code	Type	Primary Key	Mandatory
data_type_id	dataTypeld	varchar(20)	Yes	Yes
predicted_method	predictionMethod	varchar(18)	No	No
supplier_name	supplierName	varchar(30)	No	No

Table: PlDataTypeFiles

Description

Information about multi-file data types.

Column List

Name	Code	Type	Primary Key	Mandatory
datatypeid	dataTypeld	varchar(20)	Yes	Yes
filetypeid	fileTypeld	int	No	No
filenotypename	fileTypeName	varchar(40)	Yes	Yes
maxnumfiles	maxNumFiles	int	No	No

Table: PlDataTypeMaster

Description

Basic information about a data type.

Column List

Name	Code	Type	Primary Key	Mandatory
archive_center	archiveCenter	varchar(20)	No	No
catalogue_category	catalogueCategory	varchar(15)	No	No
data_type_id	dataTypeld	varchar(20)	Yes	Yes
data_type_name	dataTypeName	varchar(20)	No	No
description	dataTypeDescription	varchar(60)	No	No
DS_UR_string	dataServUrString	varchar(255)	No	No
dynamic_flag	dynamicFlag	int	No	Yes
dynamictype	dynamicType	int	No	No
hdfflag	hdfFlag	bit	No	Yes
instrument	instrument	varchar(20)	No	No

Name	Code	Type	Primary Key	Mandatory
interimlastpgeouse	interimLastPgeToUse	varchar(17)	No	No
interimlongduration	interimLongDuration	int	No	No
interimshortduration	interimShortDuration	int	No	No
maxfilesofanytype	maxFilesOfAnyType	int	No	No
maxfiletypes	maxFileTypes	int	No	No
nominal_size	nominalSize	float	No	No
platformname	platformName	varchar(25)	No	No
processing_center	processingCenter	varchar(20)	No	No
processing_level	processingLevel	varchar(6)	No	No
provider	provider	varchar(50)	No	No
QA_subscription	qaSubscription	bit	No	Yes
spatial_flag	spatialFlag	bit	No	Yes
subscription_flag	subscriptionFlag	int	No	No
version	version	varchar(20)	No	No
distParameter	distParameter	varchar(80)	No	No

Table: PIDataTypeReq

Description

Information about the association between a PGE and its input data types.

Column List

Name	Code	Type	Primary Key	Mandatory
data_type_id	dataTypeId	varchar(20)	Yes	Yes
data_type_req	dataTypeRequirement	int	No	No
endtimeoffset	endTimeOffset	integer	No	No
input_type	inputType	int	No	Yes
linkid	linkId	int	No	No
logical_id	logicalId	int	Yes	Yes
numneeded	numNeeded	int	No	No
pcfdatatype	pcfFileType	int	No	No
pge_id	pgId	varchar(17)	Yes	Yes
QA_threshold	qaThreshold	varchar(40)	No	No
sciencegroup	scienceGroup	varchar(5)	No	No
starttimeoffset	startTimeOffset	integer	No	No
whereclause	whereClause	varchar(255)	No	No
distValue	distValue	varchar(80)	No	No

Table: PlDprCompletion

Description

Statistics on the execution of a DPR entered after its completion.

Column List

Name	Code	Type	Primary Key	Mandatory
completiondate	completionDate	datetime	Yes	Yes
dprelapsedtime	dprElapsedTime	float	No	No
dprid	dprId	varchar(27)	Yes	Yes
exitcode	exitCode	int	No	No
pgeblockinputoperations	pgeBlockInputOperations	int	No	No
pgeblockoutputoperations	pgeBlockOutputOperations	int	No	No
pgecpuftime	pgeCpuTime	float	No	No
pgeelapsedtime	pgeElapsedTime	float	No	No
pgeomaxmemoryuse	pgeMaxMemoryUse	float	No	No
pgepagefaults	pgePageFaults	int	No	No
pgesharedmemoryuse	pgeSharedMemoryUse	float	No	No
pgeswaps	pgeSwaps	int	No	No
platform	platform	varchar(60)	No	No
systemcpuftime	systemCpuTime	float	No	No

Table: PlDprData

Description

Information about the relationship between a DPR and a data granule.

Column List

Name	Code	Type	Primary Key	Mandatory
accepted	accepted	int	No	Yes
datatypeRequirement	dataTypeRequirement	int	No	No
dpr_id	dprId	varchar(29)	Yes	Yes
granule_id	granuleId	varchar(30)	Yes	Yes
ioflag	ioFlag	int	Yes	Yes
linkid	linkId	int	No	No
logicalid	logicalId	int	Yes	Yes
numNeeded	numNeeded	int	No	No
order	theOrder	int	No	No
primary_type	primaryType	varchar(20)	No	No
temporal_flag	temporalFlag	bit	No	Yes
timerexp	timerExp	bit	No	Yes

Name	Code	Type	Primary Key	Mandatory
timerstart	timerStart	bit	No	Yes
timewait	timeWait	int	No	No
type	type	int	No	No
waitforflag	waitForFlag	bit	No	Yes

Table: PIesdtParam

Description

An explosion table for PIDataTypeMaster. Contains the parameter list used to retrieve the metadata associated with a newly arrived instance of the data type.

Column List

Name	Code	Type	Primary Key	Mandatory
datatypeid	dataTypeId	varchar(20)	Yes	Yes
paramname	paramName	varchar(20)	Yes	Yes
paramtype	paramType	varchar(20)	No	No
secondParm	secondParm	varchar(40)	Yes	Yes
secondValue	secondValue	varchar(80)	Yes	Yes

Table: PIesdtParamValues

Description

An explosion table for PIDataGranuleShort. Contains lists of ESDT parameter values associated with a data granule, i.e. metadata values associated with the data type used to determine the suitability of the data granule for production.

Column List

Name	Code	Type	Primary Key	Mandatory
ESDT_name	esdtParmName	varchar(40)	Yes	Yes
ESDT_parm_value	esdtParmVal	varchar(80)	No	No
granule_id	granuleId	varchar(130)	Yes	Yes
secondParm	secondParm	varchar(40)	Yes	Yes
secondValue	secondValue	varchar(80)	Yes	Yes

Table: PIFileTypeReq

Description

An explosion table for PIDataTypeReq showing the various file types associated with a data type that is input to a PGE.

Column List

Name	Code	Type	Primary Key	Mandatory
datatypeid	dataTypeId	varchar(20)	Yes	Yes
filetype	fileType	int	Yes	Yes
logicalid	logicalId	int	Yes	Yes
maxnumfiles	maxNumFiles	int	No	No
pgeid	pgId	varchar(17)	Yes	Yes

Table: PIGroundEvent

Description

Information about a ground event, i.e. the allocation of resources to some non-production task such as maintenance.

Column List

Name	Code	Type	Primary Key	Mandatory
description	groundEventDescription	varchar(40)	No	No
duration	groundEventDuration	int	No	No
ground_event_id	groundEventId	int	Yes	Yes
ground_event_name	groundEventName	varchar(40)	No	No
priority	groundEventPriority	int	No	No
win_end	groundEventEnd	datetime	No	No
win_start	groundEventStart	datetime	Yes	Yes

Table: PIGroundEventResourceExplosion

Description

A join table between PIGroundEvent and PIResource.

Column List

Name	Code	Type	Primary Key	Mandatory
ground_event_id	groundEventId	int	Yes	Yes
resourceid	resourceId	int	Yes	Yes
win_start	groundEventStart	datetime	Yes	Yes

Table: PIMetadataChecks

Description

Provides a metadata field and its corresponding value to be checked against the actual metadata of the specified input data granule when deciding if a particular input should be used or if its corresponding PGE should be executed.

Column List

Name	Code	Type	Primary Key	Mandatory
data_type_id	dataTypeId	varchar(20)	Yes	Yes
ioflag	ioFlag	int	Yes	Yes
logical_id	logicalId	int	No	Yes
metadata_oper	operator	varchar(3)	No	No
metadata_parm_name	paraName	varchar(20)	Yes	Yes
queryFlag	queryFlag	int	No	No
queryType	queryType	int	No	No
pge_id	pgId	varchar(17)	Yes	Yes
metadata_type	type	varchar(5)	No	No
metadata_value	value	varchar(40)	No	No
secondParm	secondParm	varchar(40)	Yes	Yes
secondValue	secondValue	varchar(80)	Yes	Yes

Table: PI Notification

Description

Planning table used in fault recovery.

Column List

Name	Code	Type	Primary Key	Mandatory
notificationMessage	notificationMessage	varchar(255)	Yes	Yes
processingStatus	processingStatus	int	No	Yes

Table: PI Output Data Yield

Description

Information about the output data granules associated with a PGE.

Column List

Name	Code	Type	Primary Key	Mandatory
data_type_id	dataTypeId	varchar(20)	Yes	Yes
linkid	linkId	int	No	No
logical_id	logicalId	int	Yes	Yes
max_size	maxSize	int	No	No
metadatalogid	metadataLogId	int	No	No
min_size	minSize	int	No	No
pcfdatatype	pcfFileType	int	No	No
pge_id	pgId	varchar(17)	Yes	Yes
sciencegroup	scienceGroup	varchar(5)	No	No

Name	Code	Type	Primary Key	Mandatory
yield	yield	int	No	No
distValue	distValue	varchar(80)	No	No

Table: PIOutputFileType

Description

Information about file types associated with an output data granule.

Column List

Name	Code	Type	Primary Key	Mandatory
datatypeid	dataTypeld	varchar(20)	Yes	Yes
filetype	fileType	int	Yes	Yes
logicalid	logicalld	int	Yes	Yes
maxnumfiles	maxNumFiles	int	No	No
pgeid	pgeld	varchar(17)	Yes	Yes

Table: PIPathModel

Description

Maps the current absolute path number to the instrument-specific path number.

Column List

Name	Code	Type	Primary Key	Mandatory
platform	platform	varchar(25)	Yes	Yes
pathMapName	pathMapName	varchar(25)	Yes	Yes
absolutePathNum	absolutePathNum	int	Yes	Yes
pathMapNum	pathMapNum	int	No	No

Table: PIPgeDetailParameters

Description

Details regarding the user-defined parameters associated with a PGE.

Column List

Name	Code	Type	Primary Key	Mandatory
default_value	pgeParameterDefault	varchar(200)	No	No
description	description	varchar(80)	No	No
logical_id	pgeParameterLogicalld	int	Yes	Yes
pge_id	pgeld	varchar(17)	Yes	Yes
pgeparametername	pgeParameterName	varchar(50)	No	No
pgeParametersDbQuery	pgeParametersDbQuery	int	No	No

Table: PlPgeDetailTile

Description

Details of time associated with the scheduling of a PGE.

Column List

Name	Code	Type	Primary Key	Mandatory
pge_id	pgeld	varchar(17)	Yes	Yes
schemaName	processingPeriod	varchar(20)	No	No

Table: PlPgeDetailTime

Description

Details of time associated with the scheduling of a PGE.

Column List

Name	Code	Type	Primary Key	Mandatory
orbitoffset	orbitOffset	int	No	No
pge_id	pgeld	varchar(17)	Yes	Yes
processing_boundary	processingBoundary	varchar(40)	No	No
processing_period	processingPeriod	varchar(20)	No	No

Table: PlPgeMaster

Description

Main source of information about a PGE.

Column List

Name	Code	Type	Primary Key	Mandatory
instrument	instrument	varchar(20)	No	No
minnumoutputs	minNumOutputs	int	No	No
normal_staging_time	normalStagingTime	integer	No	No
orbit_number	orbitNumber	int	No	No
pge_id	pgeld	varchar(17)	Yes	Yes
pge_name	pgeName	varchar(10)	No	Yes
pge_test_operational	pgeTestOperational	int	No	Yes
pge_version	pgeVersion	varchar(5)	No	Yes
pgecollectionid	pgeCollectionId	varchar(20)	No	No
pgescheduletype	pgeScheduleType	char(1)	No	No
platform	platform	varchar(25)	No	No
profiledescription	profileDescription	varchar(255)	No	No

Name	Code	Type	Primary Key	Mandatory
profileid	profileId	int	No	No
reprocess_staging_time	reprocessStagingTime	integer	No	No
sswid	sswid	varchar(17)	No	No
deleteFlag	deleteFlag	int	No	No
pathMapName	pathMapName	varchar(25)	No	No
checkOutputs	checkOutputs	bit	No	Yes

Table: PlPgeOrbitModel

Description

A table used to calculate the estimated times for an orbit.

Column List

Name	Code	Type	Primary Key	Mandatory
orbit_number	orbitNumber	int	Yes	Yes
orbitlength	orbitLength	int	No	No
orbitstart	orbitStart	datetime	No	No
platform	platform	varchar(25)	Yes	Yes
pathNumber	pathNumber	int	No	Yes

Table: PlPgePerformance

Description

Performance statistics for a PGE.

Column List

Name	Code	Type	Primary Key	Mandatory
block_in_operation	blockInputOperation	int	No	No
block_out_operation	blockOutputOperation	int	No	No
cpu_time	cpuTime	int	No	No
dprelapsedtime	dprElapsedTime	int	No	No
faults	faults	int	No	No
max_memory	maxMemory	float	No	No
pge_id	pgId	varchar(17)	Yes	Yes
pgeelapsedtime	pgeElapsedTime	int	No	No
run_block_in_operation	runBlockInOperation	int	No	No
run_block_out_operation	runBlockOutOperation	int	No	No
run_cpu_time	runCpuTime	int	No	No
run_max_memory	runMaxMemory	float	No	No
run_shared_memory	runSharedMemory	float	No	No
run_swaps	runSwaps	int	No	No

Name	Code	Type	Primary Key	Mandatory
rundprelapsed	runDprElapsed	int	No	No
runfaults	runFaults	int	No	No
runpgeelapsed	runPgeElapsed	int	No	No
shared_memory	sharedMemory	float	No	No
swaps	swaps	int	No	No

Table: PI_PgePriority

Description

Relates a particular PGE to a particular priority. Used by production planning to determine the priority of jobs that use that PGE.

Column List

Name	Code	Type	Primary Key	Mandatory
pgeid	pgelid	varchar(17)	Yes	Yes
priority	priority	int	No	Yes
strategyid	strategyId	varchar(20)	Yes	Yes

Table: PI_Plans

Description

Describes a production plan.

Column List

Name	Code	Type	Primary Key	Mandatory
active_status	planActiveStatus	char(1)	No	Yes
description	planDescription	varchar(255)	No	No
end_time	planCreation	datetime	No	No
plan_name	planName	varchar(20)	No	Yes
planend	planEnd	datetime	No	No
planid	planId	varchar(20)	Yes	Yes
planstart	planStart	datetime	No	No
start_time	planActivation	datetime	No	No
strategy_id	strategyId	varchar(20)	No	Yes

Table: PlPrMetadataValue

Description

Contains expected values of parameters associated with a production request.

Column List

Name	Code	Type	Primary Key	Mandatory
datatypeid	dataTypeId	varchar(20)	Yes	Yes
ioflag	ioFlag	int	Yes	Yes
metadatavalue	metadataValue	varchar(40)	No	Yes
paraname	paraName	varchar(20)	Yes	Yes
productionrequestid	productionRequestId	varchar(20)	Yes	Yes
secondParm	secondParm	varchar(40)	Yes	Yes
secondValue	secondValue	varchar(80)	Yes	Yes

Table: PlProductionRequest

Description

Information associated with a production request.

Column List

Name	Code	Type	Primary Key	Mandatory
collection_start	collectionStart	datetime	No	No
collection_stop	collectionStop	datetime	No	No
description	description	varchar(255)	No	No
num_dpr_to_keep	numDprToKeep	int	No	No
num_dpr_to_skip	numDprToSkip	int	No	No
originationdate	originationDate	datetime	No	No
saveState	saveState	char(1)	No	No
pge_id	pgelIdentifier	varchar(17)	No	Yes
pr_collection_id	prCollectionId	varchar(10)	No	No
PR_type	prType	int	No	No
priority	productionRequestPriority	int	No	No
request_id	productionRequestId	varchar(20)	Yes	Yes
requester_id	requesterId	varchar(25)	No	No
skip_first_flag	skipFirstFlag	bit	No	Yes
target_date	targetDate	datetime	No	No
target_delta	targetDelta	int	No	No
user_type	userType	varchar(20)	No	No
tileId	tileId	int	No	No

Table: PIProductionStrategy

Description

Describes a production strategy used by a DAAC to determine priorities for data processing requests.

Column List

Name	Code	Type	Primary Key	Mandatory
aging_delta	agingDelta	int	No	No
defaultpgepriority	defaultPGEPriority	int	No	No
defaultprpriority	defaultPRPriority	int	No	No
defaultuserpriority	defaultUserPriority	int	No	No
inter_DAAC_delta	interDaacDelta	int	No	No
pge_weight	pgeWeight	float	No	No
PR_type_weight	prTypeWeight	float	No	No
strategy_id	strategyId	varchar(20)	Yes	Yes
user_selected_priority_weight	userSelectedPriorityWeight	float	No	No
user_weight	userWeight	float	No	No

Table: PIPrParameter

Description

Description of a parameter associated with a production request.

Column List

Name	Code	Type	Primary Key	Mandatory
parameter_value	parameterValue	varchar(200)	No	No
pgeid	pgeId	varchar(17)	Yes	Yes
pgeparameterlogicalid	pgeParameterLogicalId	int	Yes	Yes
request_id	productionRequestId	varchar(20)	Yes	Yes

Table: PIPRPriority

Description

A list of PR-based priorities associated with a production strategy.

Column List

Name	Code	Type	Primary Key	Mandatory
priority	priority	int	No	Yes
prtype	prType	int	Yes	Yes
strategyid	strategyId	varchar(20)	Yes	Yes

Table: PIPrVsPlan

Description

A join table relating production requests to production plans.

Column List

Name	Code	Type	Primary Key	Mandatory
planid	planId	varchar(20)	No	Yes
priority	priority	int	No	No
prvsplanid	prVsPlanId	numeric(9,0)	Yes	Yes
request_id	productionRequestId	varchar(20)	No	Yes

Table: PIRescUseThresh

Description

Contains resource usage thresholds for OnDemand production.

Column List

Name	Code	Type	Primary Key	Mandatory
resc_use_thresh_id	rescUseThreshId	numeric(9,0)	Yes	Yes
resource_type	resourceType	int	No	Yes
threshold	threshold	float	No	Yes
user_type	userType	varchar(20)	No	Yes

Table: PIResource

Description

Base table for a physical resource.

Column List

Name	Code	Type	Primary Key	Mandatory
activitytypeid	activityTypeld	int	No	No
resource_id	resourceId	int	Yes	Yes
resource_name	resourceName	varchar(60)	No	Yes
resourcestate	resourceState	int	No	No
resourcetype	resourceType	varchar(15)	No	No
onLineState	onLineState	int	No	No

Table: PIResourceRequirement

Description

Describes the resource requirements of a PGE.

Column List

Name	Code	Type	Primary Key	Mandatory
computer	computer	varchar(60)	No	No
cpu	numOfCPUs	int	No	No
diskspace	diskSpace	float	No	No
exetarfilediskspace	exeTarFileDiskSpace	float	No	No
exetarur	exeTarUR	varchar(255)	No	No
exeuntarfilediskspace	exeUntarFileDiskSpace	float	No	No
mcfname	mcfName	varchar(30)	No	No
pgeid	pgeld	varchar(17)	No	No
ramsize	ramSize	float	No	No
sswid	sswId	varchar(17)	Yes	Yes
string	string	varchar(60)	No	No
toplevelshellname	topLevelShellName	varchar(30)	No	No
toolkitArchitecture	toolkitArchitecture	varchar(20)	No	Yes
pgeCommands	pgeCommands	char(4)	No	Yes

Table: PIRoutineArrival

Description

Describes the most frequent method for predicting data arrivals within ECS.

Column List

Name	Code	Type	Primary Key	Mandatory
boundary	boundary	varchar(40)	No	No
data_type_id	dataTypId	varchar(20)	Yes	Yes
delay	delay	int	No	No
period	period	varchar(20)	No	No
duration	duration	varchar(20)	No	No

Table: PIRscComputer

Description

Describes the computers in the resource configuration for the production system.

Column List

Name	Code	Type	Primary Key	Mandatory
computerid	computerId	int	Yes	Yes
computername	computerName	varchar(60)	No	No
cpu	totalCpu	int	No	No
cputallocation	cpuAllocation	int	No	No
max_disk_space	maxDiskSpace	int	No	No
operating_system	operatingSys	varchar(60)	No	No
ramallocation	ramAllocation	float	No	No
realcomputerid	realComputerId	int	No	No
stringid	stringId	int	No	No
total_ram	totalRam	int	No	No
cpuinUse	cpuInUse	int	No	No
ramInUse	ramInUse	float	No	No

Table: PlRscDiskPartition

Description

Describes the disk resources for data production.

Column List

Name	Code	Type	Primary Key	Mandatory
block_size	blockSize	int	No	No
computerid	computerId	int	No	No
device_id	partitionName	varchar(60)	No	No
diskpartitionid	diskPartitionId	int	Yes	Yes
diskusage	diskUsage	float	No	No
partition_size	partitionSize	float	No	No
sysallocation	sysAllocation	double precision	No	No
userallocation	userAllocation	double precision	No	No

Table: PlRscRealComputer

Description

Describes a real (as opposed to virtual) computer that is part of the resource configuration for production.

Column List

Name	Code	Type	Primary Key	Mandatory
realcomputerid	realComputerId	int	Yes	Yes
realcomputername	realComputerName	varchar(60)	No	No

Table: PIrscString

Description

Describes a string, i.e. a logical collection of resources allocated for processing the data collected by an instrument.

Column List

Name	Code	Type	Primary Key	Mandatory
autosysidkey	autosysIdKey	int	No	No
stringid	stringId	int	Yes	Yes
stringname	stringName	varchar(60)	No	No

Table: PITileCoordinates

Description

Contains the list of coordinates associated with a tile.

Column List

Name	Code	Type	Primary Key	Mandatory
tileId	tileId	int	Yes	Yes
tileCoordinates	tileCoordinates	varchar(255)	No	Yes
description	description	varchar(255)	No	No

Table: PITileSchemaShort

Description

Associates a tile with a schema.

Column List

Name	Code	Type	Primary Key	Mandatory
schemaName	schemaName	varchar(20)	Yes	Yes
tileId	tileId	int	Yes	Yes

Table: PlTimer

Description

Information about a timer for a data granule for which a DPR is waiting.

Column List

Name	Code	Type	Primary Key	Mandatory
dprId	dprId	varchar(29)	Yes	Yes
granuleId	granuleId	varchar(130)	Yes	Yes
logicalId	logicalId	int	Yes	Yes
timeWait	timeWait	datetime	No	Yes
timerType	timerType	int	No	Yes

Table: PlUsedByCenter

Description

An explosion table for PlDataTypeMaster, containing a list of DAACs that employ the data type.

Column List

Name	Code	Type	Primary Key	Mandatory
DAAC_name	daacName	varchar(20)	Yes	Yes
data_type_id	dataTypId	varchar(20)	Yes	Yes

Table: PlUserPriority

Description

Describes a user-based priority for a production strategy.

Column List

Name	Code	Type	Primary Key	Mandatory
priority	priority	int	No	Yes
strategyid	strategyId	varchar(20)	Yes	Yes
usertype	userType	varchar(20)	Yes	Yes

Table: RpActivityType

Description

Describes an activity type associated with a resource plan.

Column List

Name	Code	Type	Primary Key	Mandatory
activityname	activityName	varchar(20)	No	No
activitytypeid	activityTypeId	int	Yes	Yes

Table: RpReservationInterval

Description

Describes an interval used in resource planning.

Column List

Name	Code	Type	Primary Key	Mandatory
flag	flag	int	No	No
reservationid	reservationId	int	Yes	Yes
starttime	startTime	datetime	Yes	Yes
stoptime	stopTime	datetime	No	No

Table: RpResourceReservation

Description

Describes a resource reservation used in resource planning.

Column List

Name	Code	Type	Primary Key	Mandatory
activitytypeid	activityTypeId	int	No	No
actualend	actualEnd	datetime	No	No
actualstart	actualStart	datetime	No	No
description	description	varchar(30)	No	No
editeddate	editedDate	datetime	No	No
endtime	endTime	datetime	No	No
everyn	everyN	int	No	No
xclusive	exclusive	int	No	No
frequency	frequency	varchar(20)	No	No
name	name	varchar(60)	No	No
originator	originator	varchar(30)	No	No
priority	priority	int	No	No
reservationid	reservationId	int	Yes	Yes
sponsor	sponsor	varchar(30)	No	No
starttime	startTime	datetime	No	No
state	state	varchar(15)	No	No

Table: RpRscPlanComments

Description

Comments for a resource plan.

Column List

Name	Code	Type	Primary Key	Mandatory
comment	comment	varchar(255)	No	No
commentid	commentId	int	Yes	Yes
commenttype	commentType	varchar(20)	Yes	Yes
ordernum	orderNum	int	Yes	Yes

Table: RpRsvRscExplosion

Description

Associates reservation IDs with resource IDs.

Column List

Name	Code	Type	Primary Key	Mandatory
reservationId	reservationId	int	Yes	Yes
resourceId	resourceId	int	Yes	Yes

4.1.3 Columns

Brief definitions of each of the columns present in the database tables defined above are contained herein.

Column Code	Table	Description
absoluteDeletionTime	PIDataGranuleShort	Actual delete time for an interim data granule.
absolutePathNum	PIPPathModel	The path number corresponding to PIPgeOrbitModel::orbitNumber
accepted	PIDprData	Indicates whether the data granule, when available has passed metadata checks.
activityName	RpActivityType	Name of an valid activity type.
activityTypeld	RpActivityType RpResourceReservation PIResource	Identifier for a valid activity type.
actualEnd	RpResourceReservation	Actual end time of the reservation.
actualStart	PIDataProcessingRequest	The date and time when the DPR is actually processed.
actualStart	RpResourceReservation	Actual starting time of the reservation.

Column Code	Table	Description
agingDelta	PIProductionStrategy	The amount to be added to a priority for an older DPR that still has not completed.
alarmTime	PIDataProcessingRequest	Reserved for future use.
allocState	DpPrRscCpuRamAllocation	Current state of allocation.
archiveCenter	PIDataTypeMaster	DAAC where the data type is archived.
associatedScienceld	PIAssociatedScienceData	Science data identifier.
attempts	DpPrResourceLock	Number of attempts
autosysId	DpPrAutosysMapList DpPrCreationQueue PIDataProcessingRequest	The identifier of an Autosys instance.
autosysIdKey	DpPrAutosysMapList	Identifies the instance of Autosys mapped to a given physical resource.
autosysIdKey	PIRscString	Identifies the instance of Autosys that is using this string (collection of logical resources) for data processing.
auxiliaryLogicalId	PIAuxiliaryIds	Contains lists of auxiliary logical IDs associated with input granules.
AuxiliarySequenceNumber	PIAuxiliaryIds	Contains lists of auxiliary logical IDs associated with input granules.
availability	PIDataGranuleShort	Flag to indicate availability of the data.
availabilityActual	PIDataGranuleShort	The date and time when data was actually made available to the subscription manager.
availabilityPredicted	PIDataGranuleShort	The predicted date and time when data will be available to the subscription manager.
baselineTime	PIDataGranuleShort	The date and time the DPR was predicted to begin in the latest plan.
baselineTime	PIDataProcessingRequest	The availability date and time of the data granule according to the baselined active plan, agreed to by the DAACs requiring the granule.
blockInputOperation	PIPgePerformance	Number of input blocks used by the PGE during SSI&T.
blockOutputOperation	PIPgePerformance	Number of output blocks used by the PGE during SSI&T.
blockSize	PIRscDiskPartition	Block size for the partition.
boundary	PIRoutineArrival	A boundary time form which predicted times can be derived.

Column Code	Table	Description
catalogueCategory	PIDatatypeMaster	Indicates whether the data type is generated from a PGE, an intermediate file, an input file received from another site, etc.
checkOutputs	PIPgeMaster	A flag indicating that the output granules generated by this PGE require special checking.
collectionStart	PIProductionRequest	The start time, in terms of data collection, for the production request.
collectionStop	PIProductionRequest	The start time, in terms of data collection, for the production request.
comment	RpRscPlanComments	Text of the comment.
commentId	RpRscPlanComments	Comment identifier.
commentType	RpRscPlanComments	Type of comment.
completionDate	PIDprCompletion	Date DPR completed.
completionState	PIDataProcessingRequest	A status indicator describing the current status of the DPR.
computer	PIResourceRequirement	A computer within the string (collection of logical resources) required by the PGE
computerId	dpPrDiskAllocation dpPrRscCpuRamAllocation PIRscComputer PIRscDiskPartition	Identifier for the computer
computerName	PIRscComputer	Name for the computer.
containerName	PIEsdtParam	Container name.
cpuAllocation	PIRscComputer	The number of CPUs allocated.
cpuInUse	PIRscComputer	Number of processors already in use.
CpuRamAllocationCpuCount	DpPrRscCpuRamAllocation	CPU count.
cpuRamAllocationJobId	DpPrRscCpuRamAllocation	Identifier uniquely defining a particular memory allocation within a computer.
cpuRamAllocationRamSize	DpPrRscCpuRamAllocation	Amount of RAM used.
cpuTime	PIPgePerformance	Amount of CPU time used during SSI&T.
daacName	PIUsedByCenter	Name of the DAAC.
dataServUrString	PIDataTypeMaster	Universal Reference to the Data Server providing services for the data type.
dataStartTime	PIDataProcessingRequest	The start time for the data that is input to the DPR.

Column Code	Table	Description
dataStopTime	PIDataProcessingRequest	The stop time for the data that is input to the DPR.
dataType	PIAlternateInputValues	Data Type of the alternate input.
dataTypeDescription	PIDataTypeMaster	A textual description of the data type.
dataTypeId	PIAlternate PIAssociatedScienceData PIDataGranuleShort PIDataSourceMaster PIDataTypeFiles PIDataTypeMaster PIDataTypeReq PIEsdtParam PIFileTypeReq PIMetadataChecks PIOOutputDataYield PIOOutputFileType PIPPrMetadataValue PIRoutineArrival PIUsedByCenter	Identifier for the data type.
dataTypeName	PIDataTypeMaster	The short name, descriptive acronym, of a data type
dataTypeRequirement	PIDataTypeReq PIDprData	
DateLastAccessed	DpPrFile	Date and time when file was last accessed.
defaultOrder	PIAlternate	Defaule priority of an alternate input earth science data type.
defaultPGEPriority	PIProductionStrategy	Default value for the PGE-based priority.
defaultPRPriority	PIProductionStrategy	Default value for the PR-based priority.
defaultTimer	PIAlternate	The default length of time that the Subscription Manager will wait for an alternate input earth science data type to arrive
defaultUserPriority	PIProductionStrategy	Default value for the user-based priority.
delay	PIRoutineArrival	The average delay between data collection and the arrival in ECS of the corresponding data granule.
delayBeforeQuery	PIPgeMaster	Minimum wait in seconds after the data collection stop time before a query is done on the data server for the input data.

Column Code	Table	Description
deleteFlag	PIPgeMaster	Indicates that the PGE is waiting for notification from the Subscription Manager that input data is available.
dependentDprId	DpPrDprDependList	Identifier for a dependent DPR.
dependencySswId	DpPrPgeExitDependency	Science software ID of a dependent PGE.
description	PITileCoordinates	Description of the tile.
description	RpResourceReservation	Description of the reservation.
description	PIPgeDetailParameters	A description of the user parameter.
description	PIProductionRequest	A textual description of the production request.
diskAllocationActual	DpPrDiskAllocation	The actual amount of disk space being consumed by the allocated file.
diskAllocationId	DpPrDiskAllocation	Internal identifier to uniquely track disk allocations.
diskAllocationSize	DpPrDiskAllocation	The amount of disk space specified for the original allocation request.
diskAllocationType	DpPrDiskAllocation	The type of allocation: system files or user (science software) files.
diskPartitionId	DpPrDiskAllocation	Identifier of a disk partition.
diskSpace	PIResourceRequirement	The amount of disk space required for a PGE.
diskUsage	PIRscDiskPartition	Current amount of disk resources allocated for science processing
distParameter	PIDataTypeMaster	Distinguishing parameter.
distValue	PIDataTypeReq PIOOutputDataYield	Distinguishing value.
dprCollectionId	PIDDataProcessingRequest	The collection identifier, if this DPR is part of a collection.
dprElapsedTime	PIDprCompletion PIPgePerformance	Elapsed time of DPR execution.
dprId	DpPrCerationQueue DpPrDprDependList DpPrGranuleLocation DpPrPcf DpPrPge PIDDataProcessingRequest PIDprCompletion PIDprData PITimer	Identifier of the DPR.
duration	PIRoutineArrival	Duration of data arrival.

Column Code	Table	Description
dynamicFlag	PIDataTypeMaster	Tells whether the earth science data type is static or dynamic.
dynamicType	PIDataTypeMaster	Indicates the type of dynamic data.
ecsUnit	DpPrResourceLock	ECS identification number.
editedDate	RpResourceReservation	Date on which reservation was last edited.
endTime	RpResourceReservation	End time of the reservation.
endTimeOffset	PIDataTypeReq	Number of seconds to add or subtract from the endtime when acquiring this input.
esdtParmName	PIEsdtParamValues	Name of the earth science data type parameter.
esdtParmVal	PIEsdtParamValues	The earth science data type parameter value.
everyN	RpResourceReservation	Reserved for future use.
exclusive	RpResourceReservation	Indicates whether the reservation is exclusive.
execLayer	DpPrExecutable	Layer of execution.
execLocation	DpPrDprDependList	The disk location of the executable.
execMachine	DpPrDprDependList DpPrPge	The machine on which a job execution is performed.
execName	DpPrDprDependList DpPrPge	The name of the executable file.
execPermission	DpPrDprDependList	The file system permissions setting for the executable.
execShell	DpPrDprDependList	For executables which are shell scripts, the type of shell for which the script was written.
exeTarFileDiskSpace	PIResourceRequirement	Disk space required for the tar file containing the executable.
exeTarUR	PIResourceRequirement	Universal reference for the tar image containing the executable.
exeUntarFileDiskSpace	PIResourceRequirement	Disk space required for the executable after it has been extracted from the tar file.
exitCode	DpPrPgeExitDependency DpPrPgeExitMessage PIDprCompletion	Final exit condition returned by the PGE to the processing system.
exitMessage	DpPrPgeExitMessage	Text of message upon termination.
exitOperation	DpPrPgeExitDependency	Description of the exit operation.
faults	PIPgePerformance	Number of page faults during SSI&T.
fileId	DpPrFile	A numerical identifier of a file containing (part of) a data granule.

Column Code	Table	Description
fileName	DpPrFile	The name of a data granule file.
fileNumber	DpPrFile	Number of a file (if part of a multi-file granule).
fileSize	DpPrFile	Size of a file
fileType	DpPrFile PIFileTypeReq PIOutputFileType	Type of file.
fileTypeld	PIDataTypeFiles	Numerical identifier for a file type.
fileTypeName	PIDataTypeFiles	Name of a file type associated with the data type.
flag	RpReservationInterval	Reserved for future use.
frequency	RpResourceReservation	Frequency of the reservation.
granuleId	DpPrGranuleLocation PIDataGranuleShort PIDprData PIEsdtParamValues PITimer	Identifier of a data granule.
granuleSize	PIDataGranuleShort	The storage requirements of the data granule.
groundEventActivation	PIGroundEvent	Start date and time for a ground event.
groundEventDescription	PIGroundEvent	Description of a ground event.
groundEventDuration	PIGroundEvent	The duration of a ground event.
groundEventEnd	PIGroundEvent	End date and time for a ground event.
groundEventId	PIGroundEvent	Identifier of the ground event.
groundEventName	PIGroundEvent	Name for a ground event.
groundEventPriority	PIGroundEvent	The priority of the ground event.
groundEventStart	PIGroundEvent	Starting date and time for a ground event.
hdfFlag	PIDataTypeMaster	Indicates whether this data type contains HDF data.
hold	DpPrCreationQueue	Flag indicating that Autosys has this DPR on hold.
inputType	PIDataTypeReq	Type of input (required, primary, or backup).
instrument	PIDataTypeMaster PIPgeMaster	The short name or acronym by which an instrument is known.
interDaacDelta	PIProductionStrategy	Amount to add to the calculated priority of an activity if the DPR produces data that is needed at a remote DAAC.
interimLastPageToUse	PIDataTypeMaster	ID of the last PGE to use this interim data type.

Column Code	Table	Description
interimLongDuration	PIDataTypeMaster	Longest duration this interim data should last until it is deleted from the data server.
interimShortDuration	PIDataTypeMaster	Shortest duration this interim data could last before it is deleted from the data server.
ioFlag	PIDprData	ioFlag indicates whether metatdat check should be performed on input or output.
ioFlag	PIMetadataChecks	Used to distinguish between input and output parameters.
ioFlag	PIPrMetadataValue	Indicates whether the granule is input or output with respect to this DPR.
jobId	DpPrResourceLock	Identifier of the current job.
keyLogicalId	PIPgeMaster	Indicates which input in the list of inputs for this PGE should be treated as the key when performing a spatial query.
lateMessageSent	PIDataProcessingRequest	Reserved for future use.
linkId	PIDataTypeReq	Reserved for future use.
linkId	PIDprData	Reserved for future use
linkId	PIOOutputDataYield	Indicates which instance of data is to be assigned to the ID.
logicalId	PIAlternate PIAlternateInputValues PIAssociatedScienceData	Identifier to distinguish among multiple instances of a PGE/dataType combination.
logicalId	PIDataTypeReq	The SDP toolkit logical identifier through which the data type is referenced.
logicalId	PIDprData	Identifies which multiple instances of DPR/granule pairs is meant.
logicalId	PIFileTypeReq	Used to distinguish among multiple granules of the same data type and associated PGE.
logicalId	PIMetadataChecks PITimer	Identifier which singles out an instance of a (DPR, data granule) pairing.
logicalId	PIOOutputDataYield	Numerical identifier for distinguishing among multiple alternate inputs having the same data type and referring to the same PGE.
logicalId	PIOOutputFileType	The SDP toolkit through which the data type is referenced.

Column Code	Table	Description
logicalId	PIPgeDetailParameters	Used to distinguish among multiple instances of a PGE/data type pair.
machineId	DpPrFile DpPrGranuleLocation	Name or identifier of a machine where a data granule or file resides.
maxDiskSpace	PIRscComputer	Maximum disk space (storage) on this computer.
maxFileOfAnyType	PIDataTypeMaster	The maximum number of files corresponding to this data type, regardless of file type.
maxGranReq	PIDataTypeReq	Maximum number of granules of this input type.
maxGranYield	PIOOutputDataYield	The number of data granules produced from the DPR.
maxMemory	PIPgePerformance	Maximum amount of memory required by the PGE during SSI&T.
maxNumFiles	PIDataTypeFiles PIFileTypeReq PIOFileType	The maximum number of files for a particular file type.
maxSize	PIOOutputDataYield	Maximum size of the output data granule.
mcfName	PIResourceRequirement	Name of the metadata control file associated with the PGE.
metadataLogId	PIOOutputDataYield	Logical ID of the MCF for this output.
metadataValue	PIPrMetadataValue	The metadata value.
minGranReq	PIDprData PIDataTypeReq	Minimum number of granules of this input type required by the PGER when forming a DPR.
minGranYield	PIOOutputDataYield	Minimum number of output granules produced.
minNumOutputs	PIPgeMaster	The minimum number of data granules the PGE PGE should generate.
minSize	PIOOutputDataYield	Minimum size of the output data granule.
name	PIDataProcessingRequest	Name for reservation.
name	RpResourceReservation	Same as the DPR identifier.
nominalSize	PIDataTypeMaster	The nominal size of the data type.
normalStagingTime	PIPgeMaster	The time normally required to stage (make local) the input data for the PGE.
notificationMessage	PINotification	Notification message.
numberOfUsage	DpPrFile	How many DPRs use this file.

Column Code	Table	Description
numDprToKeep	PIProductionRequest	For intermittently activated PGEs, the number of DPRs to be kept for this PR.
numDprToSkip	PIProductionRequest	For intermittently activated DPRs, the number of DPRs to skip (after the number to keep have been created).
numNeeded	PIDataTypeReq	The number of inputs granules required by the DPR.
numNeeded	PIDprData	The number of inputs required by the PGE.
object	DpPrRpclId	Object name.
onLineState	PIResource	Indicates whether the resource is online or offline.
operatingSys	PIRscComputer	The operating system name and version running on this computer.
operatingSystem	DpPrDprDependList PIRscComputer	Target operating system for the executable.
operator	PIMetadataChecks	The operation (=,>,<) for the comparison of the metadata field.
orbitLength	PIPgeOrbitModel	The length of the orbit.
orbitNumber	PIPgeOrbitModel	The number of the orbit.
orbitOffset	PIPgeDetailTime	Orbit offset.
orbitStart	PIPgeOrbitModel	The starting time of the orbit.
orderNum	RpRscPlanComments	Order number associated with the production request.
originationDate	PIProductionRequest	Date and time when the production request was entered.
originator	RpResourceReservation	Originator of the reservation.
parameterValue	PIPrParameter	Value for the parameter.
paramName	PIEsdtParam	The parameter name.
paramType	PIEsdtParam	The parameter type.
paraName	PIMetadataChecks PIPrMetadataValue	The name of the parameter (metadata field).
partitionName	PIRscDiskPartition	Name for the partition.
partitionSize	PIRscDiskPartition	Total size of the partition.
path	DpPrDiskAllocation DpPrFile	Location of the file as a path name.
pathMapName	PIPPathModel	Name of the mapping.
pathMapName	PIPgeMaster	Mapping name used to identify the path number.
pathMapNum	PIPPathModel	Mapping from the absolute path number.

Column Code	Table	Description
pathNumber	PIPgeOrbitModel	Similar to the absolute orbitNumber field, but refers to the 1-233 repeating orbit cycle inherent in the AM1 spacecraft.
pcfFileType	PIOOutputDataYield	Provides information to DPS as to which portion of the PCF the data file is referenced from in the SDP toolkit.
pcfFileType	PIDataTypeReq PIOOutputDataYield	Indicates the portion of the Process Control File from which the data file is referenced in the SDP toolkit.
pcfLocation	DpPrPcf	Pathname fo the directory containg the PCF.
pcfName	DpPrPcf	Filename of the PCF.
pcfPermission	DpPrPcf	File permissions that need to be placed on the PCF.
pcfType	DpPrPcf	Indicates the type of PCF.
period	PIRoutineArrival	The time period used in predicting arrivals (hour, day, set of orbits, etc.)
pgeBlockInputOperations	PIDprCompletion	Number of block input operations.
pgeBlockOutputOperations	PIDprCompletion	Number of block output operations.
pgeCollectionId	PIPgeMaster	If the PGE belongs to a collection, the identifier for that collection.
pgeCommands	DpPrPge PIResourceRequirement	Startup condition information for the activation shell.
pgeCpu	DpPrPge	CPU associated with the PGE.
pgeCpuTime	PIDprCompletion	Amount of CPU time used by the PGE.
pgeElapsedTime	PIDprCompletion PIPgePerformance	Elapsed time of PGE execution.
pgeEnvironment	DpPrPge	Environment variables that need to be set prior to running the PGE.

Column Code	Table	Description
pgeId	PIAlternate PIAssociatedScienceData PIAuxiliaryLogicalId PIDataProcessingRequest PIDataTypeReq PIFileTypeReq PIMetadataChecks PIOOutputDataYield PIOOutputFileType PIPgeDetailParameters PIPgeDetailTile PIPgeDetailTime PIPgeMaster PIPgePerformance PIPgePriority PIProductionRequest PIPrParameter PIResourceRequirement	Unique identifier for the PGE.
pgeIdentifier	PIProductionRequest	Identifier for the associated PGE.
pgeMaxMemoryUse	PIDprCompletion	Maximum memory used by the PGE.
pgeName	PIPgeMaster	Name for the PGE.
pgePageFaults	PIDprCompletion	Number of page faults during PGE execution.
pgeParameterDefault	PIPgeDetailParameters	Default value for the user parameter.
pgeParameterLogicalId	PIPrParameter	ID of the parameter for this PGE.
pgeParameterLogicalId	PIPrParameter	Parameter number for the PGE.
pgeParameterName	PIPgeDetailParameters	Name of the user parameter.
pgeParametersDbQuery	PIPgeDetailParameters	Indicates whether a PGE parameter needs to look in the database for a value.
pgeScheduleType	PIPgeMaster	Indicates whether the PGE will be scheduled by time, by data arrival, or by some other method.
pgeSharedMemoryUse	PIDprCompletion	Amount of shared memory used by the PGE.
pgeShell	DpPrPge	The processing shell which activates the science software's outer shell.
pgeSswId	DpPrPgeExitDependency	Science software ID associated with the PGE.

Column Code	Table	Description
pgeSswId	DpPrPgeExitMessage	Science software ID associated with the PGE.
pgeState	DpPrPge	Current state of the PGE.
pgeSwaps	PIDprCompletion	Number of swaps during PGE execution.
pgeTestOperational	PIPgeMaster	Indicates if the PGE's status is test or operational.
pgeVersion	PIPgeMaster	Version number of the PGE.
pgeWeight	PIProductionStrategy	The weight given to the priority, based on the PGE.
pid	DpPrResourceLock	Process identifier.
planActivation	PIPlans	The start date and time for the data granule.
planActiveStatus	PIPlans	Indicates whether the plan is currently active.
planCreation	PIPlans	When the plan was created.
planDescription	PIPlans	Description of the purpose of the plan.
planEnd	PIPlans	Ending point of the time frame for the plan.
planId	PIPlans PIPrVsPlan	Identifier for a production plan.
planName	PIPlans	Name for the plan.
planStart	PIPlans	Starting point of the time frame for the plan.
platform	PIDprCompletion	Actual science processing hardware used for this execution.
platform	PIPathModel PIPgeMaster PIPgeOrbitModel	The applicable spacecraft/satellite.
platformName	PIDataTypeMaster	The spacecraft/satellite associated with the instrument.
prCollectionId	PIProductionRequest	Identifier for a collection, if PR belongs to a collection.
predictedStagingTime	PIDataGranuleShort	An estimate of the length of time that it will take to stage this data granule prior to data processing.
predictedStart	PIDataProcessingRequest	The predicted date and time when this DPR will be run by Data Processing.
predictionMethod	PIDataSourceMaster	The method by which data availability prediction occurs (e.g. routine arrival, FOS-based).
primaryLogicalId	PIAuxiliaryLogicalIds	

Column Code	Table	Description
primaryType	PIAlternate	The data type of the primary alternate input associated with this data granule.
primaryType	PIAlternateInputValues	A reference to the primary input for which this input is alternate.
primaryType	PIDprData	ID of the primary data type for this alternate input.
priority	DpPrCreationQueue	The numerical priority assigned to this DPR (usually inherited from the PR).
priority	DpPrResourceLock	The priority associated with the production request type.
priority	PIPgePriority	The priority associated with the PGE.
priority	PIPrVsPlan	The priority of the production request within the production plan.
priority	PIDataProcessingRequest PIProductionRequest	Priority for this DPR.
priority	PIPRPriority	Priority associated with a production strategy..
priority	PIUserPriority	The user-based priority.
priority	RpResourceReservation	Priority of the current job.
processingBoundary	PIPgeDetailTime	Identifies the time needed for data for extrapolation purposes.
processingCenter	PIDataTypeMaster	The name of the DAAC that produces this data type.
processingLevel	PIDataTypeMaster	Indicates processing level of the data (L0, for example).
processingPeriod	PIPgeDetailTile PIPgeDetailTime	Identifies the acquisition period for the input data.
processingStatus	PINotification	Processing status.
productionRequestId	PIAlternateInputValues PIPrParameter	Identifier for the production request.
productionRequestPriority	PIProductionRequest	The priority requested by the user when submitting the production request.
productionRequestId	PIDataProcessingRequest PIPrMetadataValue PIProductionRequest PIPrVsPlan	Identifier for the production request.
profileDescription	PIPgeMaster	The pathname of the PGE profile.
profileId	PIPgeMaster	An identifier for the profile information associated with the PGE.

Column Code	Table	Description
provider	PIDataTypeMaster	The DAAC that is maintaining the data.
prType	PIProductionRequest	Indicates the production type: routine, on-demand, or reprocessing.
prType	PIPrParameter	The production request type: standard, on-demand, or reprocessing.
prTypeWeight	PIProductionStrategy	The weight given to the priority, based on the production request type.
prVsPlanId	PIPrVsPlan	An identity for this table.
qaSubscription	PIDataTypeMaster	Flag indicating whether a subscription has been set up for the QA of this data type.
qaThreshold	PIDataTypeReq	The quality threshold to be applied to the data granules as a test of their suitability as input to the PGE.
queryFlag	PIMetadataChecks	Indicates whether metadata is query-based.
queryType	PIDataTypeReq PIMetadataChecks	Defines what type of query will be performed on this input.
queryType	PIMetadataChecks	Type of query to be performed.
ramAllocation	PIRscComputer	Available RAM.
ramInUse	PIRscComputer	Amount of RAM marked as currently in use and hence unavailable for other processes.
ramSize	PIResourceRequirement	Amount of RAM required for the PGE.
readableTag	DpPrRpclD	Readable tag.
realComputerId	PIRscComputer PIRscPIRscRealComputer	Identifier for the real as opposed to virtual computer.
realComputerName	PIRscPIRscRealComputer	Name of the real as opposed to virtual computer.
reprocessStagingTime	PIPgeMaster	The staging time normally required for reprocessing.
requesterId	PIProductionRequest	Identifies the person entering the production request.
rescUseThreshld	PIRescUseThresh	Identity for this table.
reservationId	RpReservationInterval RpResourceReservation	Identifier for a resource reservation.
resourceId	PIGroundEvent	Numeric identifier for the resource.
resourceId	PIResource	Identifier for the resource.
resourceName	PIResource	Name for the resource.
resourceState	PIResource	Current state of the resource.

Column Code	Table	Description
resourceString	DpPrAutoSysMapList	Name of computer string employed by this instance of Autosys
resourceType	PIRescUseThresh PIResource	Type of resource.
resourceType	PIRescUseThresh PIResource	Type of resource.
runBlockInOperation	PIPgePerformance	Number of input blocks required during production.
runBlockOutOperation	PIPgePerformance	Number of output blocks required during production.
runCpuTime	PIPgePerformance	Amount of CPU time required by the PGE during production.
runDprElapsed	PIPgePerformance	Elapsed time required by the DPR during production.
runFaults	PIPgePerformance	Number of page faults during production.
runMaxMemory	PIPgePerformance	Maximum amount of memory required by the PGE during production.
runPgeElapsed	PIPgePerformance	Elapsed time required by the PGE during production.
runSharedMemory	PIPgePerformance	Amount of shared memory used during production.
runSwaps	PIPgePerformance	Number of swaps during production.
runTimelD	PIAlternate	Logical ID of the PCF entry which tells the PGE which alternate was chosen.
saveState	PIProductionRequest	Saves the current state of the production request editor.
schemaName	PITileSchemaShort	Name for the tile schema.
scienceGroup	PIDataGranuleShort PIDataTypeReq PIOOutputDataYield	Describes the data group that a file or granule is associated with, science data, browse, QA.
secondParm	PIEsdtParm PIEsdtParmValues PIMetadataChecks PIPrMetadataValue	Identifier of the second metadata parameter.
secondValue	PIEsdtParm PIEsdtParmValues PIMetadataChecks PIPrMetadataValue	Value of a second metadata parameter.
sequenceNumber	DpPrFile	Sequence number of the data granule file.

Column Code	Table	Description
sharedMemory	PIPgePerformance	Amount of shared memory used during SSI&T.
skipFirstFlag	PIProductionRequest	Indicates whether we skip DPR first or keep DPR first.
spatialFlag	PIDataTypeMaster	Flag indicating spatial data, i.e. data that requires spatial coordinates in addition to start and stop times.
sponsor	RpResourceReservation	Sponsor for the reservation.
sswId	DpPrDprDependList DpPrPge PIDataProcessingRequest PIPgeMaster PIResourceRequirement	Science software identifier.
stageState	PIDataGranuleShort	Indicates the progress towards staging the file locally.
stageState	DpPrGranuleLocation	Location state of the data granule: staging, local, destaging, etc.
startTime	PIDataGranuleShort	The start date and time for the data granule.
startTime	RpReservationInterval	Start time for the interval.
startTime	RpResourceReservation	Start time of the reservation.
startTimeOffset	PIDataTypeReq	Number of seconds to add or subtract from the start time when acquiring this input.
state	DpPrResourceLock	Current state of a data processing job.
state	RpResourceReservation	Current state of the reservation.
stopTime	RpReservationInterval	The stop date and time for the data granule.
stopTime	RpResourceReservation	Stop time for the interval.
strategyId	PIPgePriority PIPlans PIProductionStrategy PIPrParameter PIUserPriority	Identifier of the associated production strategy.
string	PIResourceRequirement	The string (collection of logical resources) required by the PGE.
stringId	PIRscComputer PIRscString	Identifier for the string (collection logical of resources) containing the computer.
stringName	PIRscString	Name for the string.
subscriptionFlag	PIDataTypeMaster	Flag indicating whether Planning currently holds a subscription on this data type.

Column Code	Table	Description
supplierName	PIDDataSourceMaster	Name of the supplier of the data.
swaps	PIPgePerformance	Number of swaps during SSI&T.
sysAllocation	PIRscDiskPartition	Size within the partition allocated for system usage.
systemCpuTime	PIDprCompletion	Amount of system time spent during execution.
targetDate	PIProductionRequest	The desired time for completion of the production request.
targetDelta	PIProductionRequest	Elapsed time from the original target date.
temporalFlag	PIAlternate PIAlternateInputValues PIDprData	A flag indicating whether the most recent alternate input can be used if a primary granule of the same data type cannot be found for the production request time frame.
theOrder	PIAlternateInputValues	The relative priority of an alternate input data granule.
theOrder	PIDprData	Order number associated with production request.
threshold	PIRescUseThresh	The resource usage threshold for the particular resource and user types.
tileCoordinates	PITileCoordinates	Coordinates defining the tile.
tileId	PIDataGranuleShort PIDataProcessingRequest PIProductionRequest PITileCoordinates PITileSchemaShort	For data granules that are geographical tiles rather than time-continuous data, the identifier of the tile within the start and stop time of the orbit set.
timeDelta	PIDataTypeReq	Length of time we must spend waiting for the data to arrive.
timerExp	PIDprData	Indicates whether the timer has expired.
timerStart	PIDprData	Indicates whether the timer has started.
timerType	PITimer	Indicates the type of timer, for use by Subscription Manager.
timerWait	PIAlternateInputValues PIDprData	Length of time to wait for this data type.
timeStamp	PIDataGranuleShort PIDataProcessingRequest	The date and time this table row was last modified.
timeWait	PITimer PIDprData	The associated timer.
toolkitArchitecture	PIResourceRequirement	Which toolkit architecture was used to generate the PGE.

Column Code	Table	Description
topLevelShellName	PIResourceRequirement	Description of the shell in which the PGE is executed.
totalCpu	PIRscComputer	Total number of CPUs for this computer.
totalRam	PIRscComputer	Total RAM for this computer.
type	PIAlternateInputValues	Alternate input type (required/primary/backup).
type	PIDprData PIMetadataChecks	The type (int, string, etc.) of the metadata parameter to be compared.
universalReference	DpPrFile	Location of the file as a Universal Reference.
universalReference	PIDataGranuleShort	The Universal Reference associated with this data granule.
userAllocation	PIRscDiskPartition	Size within the partition allocated for production usage.
userSelectedPriorityWeight	PIProductionStrategy	The weight assigned to the priority by the user.
userType	PIProductionRequest	The type of user submitting the production request.
userType	PIRescUseThresh	Type of user.
userType	PIUserPriority	Type of user.
userWeight	PIProductionStrategy	The weight given to the priority, based on the user.
value	PIMetadataChecks	The value for the metadata parameter to be compared with the actual metadata value.
version	PIDataGranuleShort	Used to distinguish between two granules having the same data type and start time as the result of reprocessing.
version	PIDataTypeMaster	Version of the data type.
waitForFlag	PIAlternate	A flag indicating whether Planning should wait for this alternate input after the timer has expired. If FALSE, then Planning will permit the PGE to run without this input.
waitForFlag	PIDprData	Indicates whether DPR should be released after last timer, even if the chain is incomplete.
whereClause	PIDataTypeReq	An alternative search criterion for selecting input data associated with a DPR.

4.1.4 Column Domains

Domains specify the ranges of values allowed for a given table column. Sybase supports the definition of specific domains to further limit the format of data for a given column. Sybase domains are, in effect, user-defined data types. There are no domains defined in the PDPS database.

4.1.5 Rules

Sybase supports the definitions of rules. Rules provide a means for enforcing domain constraints on a given column. All rules defined in Sybase for the PDPS database are described herein.

Rules

Column	Rule
resourceType	= 0, 1, 2, or 3
pathNumber	>= 0
userWeight	between 0 and 100
prTypeWeight	between 0 and 100
agingDelta	between 0 and 100
interDaacDelta	between 0 and 100
userSelectedPriorityWeight	between 0 and 100
defaultPGEPriority	between 0 and 10
defaultPRPriority	between 0 and 10
defaultUserPriority	between 0 and 10
pgeTestOperational	= 0 or 1
inputType	= 0, 1, or 2
prType	= 0, 1, or 2
pctType	= 0, 1, or 2

4.1.6 Defaults

Defaults are used to supply a value for a column when one is not defined at insert time. All defaults defined in Sybase for the PDPS database are described herein.

Defaults

Table	Column	Value
DpPrExecutable	execPermission	'500'
DpPrExecutable	execShell	'csh'
DpPrPcf	pcfPermission	'600'
DpPrPcf	pcfType	0
DpPrPge	pgeCommands	'1011 50'
DpPrPge	pgeShell	'PGS_PC_Shell.sh'
DpPrRscCpuRamAllocation	allocState	0
PIDataGranuleShort	timeStamp	getdate()

Table	Column	Value
PIDataProcessing	tileId	0
PIDataProcessing	timeStamp	getdate()
PIDataTypeMaster	subscriptionFlag	0
PIPgeMaster	delayBeforeQuery	0
PIPgeOrbitMaster	pathNumber	0
PIResource	onLineState	0
PIRscComputer	cpulnUse	0
PIRscComputer	ramlnUse	0.0

4.1.7 Views

Sybase allows the definition of views as a means of limiting an application or users access to data in a table or tables. Views create a logical table from columns found in one or more tables. All views defined in the PDPS databases are described herein.

View List

Name	Code	Upd	Gen
PIDataGranule	PIDataGranule	Yes	Yes
PITileSchema	PITileSchema	No	Yes

View: PIDataGranule

Code

```
/*
=====
/* View: PIDataGranule */
=====
create view PIDataGranule as
select PIDataGranuleShort.granuleId, PIDataGranuleShort.dataTypeId,
PIDataGranuleShort.universalReference, PIDataGranuleShort.startTime,
PIDataGranuleShort.stopTime, PIDataGranuleShort.availability,
PIDataGranuleShort.availabilityActual, PIDataGranuleShort.availabilityPredicted,
PIDataGranuleShort.baselineTime, PIDataGranuleShort.granuleSize,
PIDataGranuleShort.scienceGroup, PIDataGranuleShort.tileId,
PIDataGranuleShort.predictedStagingTime, PIDataGranuleShort.absoluteDeletionTime,
PIDataGranuleShort.stageState, PIDataTypeMaster.dynamicFlag,
PIDataTypeMaster.interimLastPgeToUse, PIDataTypeMaster.interimLongDuration,
PIDataTypeMaster.interimShortDuration, PIDataTypeMaster.processingLevel,
PIDataTypeMaster.maxFileTypes, PIDataTypeMaster.maxFilesOfType,
PIDataTypeMaster.hdfFlag, PIDataGranuleShort.timeStamp
from PIDataGranuleShort, PIDataTypeMaster
where PIDataGranuleShort.dataTypeId = PIDataTypeMaster.dataTypeId
*/

```

```

/* View: PItileSchema
 */
create view PItileSchema as
select PItileCoordinates.tileId, PItileCoordinates.tileCoordinates,
PItileSchemaShort.schemaName from PItileCoordinates, PItileSchemaShort where
PItileCoordinates.tileId = PItileSchemaShort.tileId

```

4.1.8 Integrity Constraints

Sybase allows the enforcement of referential integrity via the use of declarative integrity constraints. Integrity constraints allow the SQL server to enforce primary and foreign key integrity checks automatically without requiring programming. Sybase is only ANSI-92 compliant, however, therefore its constraints support “restrict-only” operations. This means that a row cannot be deleted or updated if there are rows in other tables having a foreign key dependency on that row. Cascade delete and update operations cannot be performed if a declarative constraint has been used. In the PDPS database, since cascades of delete and update operations are desired, referential integrity is enforced via triggers rather than by declarative integrity constraints.

Dependencies on Table: DpPrAutosysMapList

Reference by List

Referenced by	Primary Key	Foreign Key
PIRscString	autosysIdKey	autosysIdKey

Dependencies on Table: DpPrCreationQueue

Reference by List

Referenced by	Primary Key	Foreign Key
DpPrDprDependList	dprId	dprId

Dependencies on Table: DpPrExecutable

Reference by List

Referenced by	Primary Key	Foreign Key
DpPrPge	sswId execName execMachine	sswId execName execMachine

Dependencies on Table: DpPrPge

Reference by List

Referenced by	Primary Key	Foreign Key
DpPrPcf	dprId	dprId

Dependencies on Table: PIDataGranuleShort

Reference by List

Referenced by	Primary Key	Foreign Key
PIEsdtParmValues	granuleId	granuleId
PIDprData	granuleId	granuleId
DpPrGranuleLocation	granuleId	granuleId

Dependencies on Table: PIDataProcessingRequest

Reference by List

Referenced by	Primary Key	Foreign Key
PIDprData	dprId	dprId
PIDprCompletion	dprId	dprId

Dependencies on Table: PIDataSourceMaster

Reference by List

Referenced by	Primary Key	Foreign Key
PIRoutineArrival	dataTypeId	dataTypeId

Dependencies on Table: PIDataTypeMaster

Reference by List

Referenced by	Primary Key	Foreign Key
PIDataGranuleShort	dataTypeId	dataTypeId
PIOOutputDataYield	dataTypeId	dataTypeId
PIDataTypeReq	dataTypeId	dataTypeId
PIEsdtParam	dataTypeId	dataTypeId
PIUsedByCenter	dataTypeId	dataTypeId
PIDataTypeFiles	dataTypeId	dataTypeId
PIDataSourceMaster	dataTypeId	dataTypeId

Dependencies on Table: PIDataTypeReq

Reference by List

Referenced by	Primary Key	Foreign Key
PIAlternate	pgeld	pgeld
PIMetadataChecks	logicalId	logicalId
PIFileTypeReq	pgeld logicalId	pgeld logicalId

Dependencies on Table: PIGroundEvent

Reference by List

Referenced by	Primary Key	Foreign Key
PIGroundEventResourceExplosion	groundEventId groundEventStart	groundEventId groundEventStart

Dependencies on Table: PIOOutputDataYield

Reference by List

Referenced by	Primary Key	Foreign Key
PIMetadataChecks	dataTypeId	dataTypeId
PIOOutputFileType	pgeld dataTypeId logicalId	pgeld dataTypeId logicalId

Dependencies on Table: PIPgeDetailParameters

Reference by List

Referenced by	Primary Key	Foreign Key
PIPParameter	pgeParameterLogicalId	pgeParameterLogicalId

Dependencies on Table: PIPgeMaster

Reference by List

Referenced by	Primary Key	Foreign Key
PIOOutputDataYield	pgeld	pgeld
PIDataTypeReq	pgeld	pgeld
PIPgeDetailTime	pgeld	pgeld
PIDataProcessingRequest	pgeld	pgeld
PIPProductionRequest	pgeld	pgelIdentifier
PIPgeDetailTile	pgeld	pgeld
PIPgePerformance	pgeld	pgeld

Referenced by	Primary Key	Foreign Key
PIPgeDetailParameters	pgeld	pgeld
PIPProductionRequest	pgeld	pgeldidentifier

Dependencies on Table: PIPgeOrbitModel

Reference by List

Referenced by	Primary Key	Foreign Key
PIPgeMaster	platform	platform

Dependencies on Table: PIPlans

Reference by List

Referenced by	Primary Key	Foreign Key
PIPrVsPlan	planId	planId

Dependencies on Table: PIProductionRequest

Reference by List

Referenced by	Primary Key	Foreign Key
PIPrMetadataValue	productionRequestId	productionRequestId
PIAlternatInputValues	productionRequestId	productionRequestId
PIPrParameter	productionRequestId	productionRequestId
PIDataProcessingRequest	productionRequestId	productionRequestId
PIPrVsPlan	productionRequestId	productionRequestId

Dependencies on Table: PIProductionStrategy

Reference by List

Referenced by	Primary Key	Foreign Key
PIPgePriority	strategyId	strategyId
PIUserPriority	strategyId	strategyId
PIPRPriority	strategyId	strategyId
PIPlans	strategyId	strategyId

Dependencies on Table: PIResource

Reference by List

Referenced by	Primary Key	Foreign Key
PIRscComputer	resourceId	computerId
PIGroundEventResourceExplosion	resourceId	resourceId
RpRsvRscExplosion	resourceId	resourceId

Referenced by	Primary Key	Foreign Key
PIRscRealComputer	resourceId	realComputerId
DpPrAutosysMapList	resourceId	autosysIdKey
PIRscDiskPartition	resourceId	diskPartitionId
PIRscString	resourceId	stringId

Dependencies on Table: PIResourceRequirement

Reference by List

Referenced by	Primary Key	Foreign Key
DpPrPgeExitDependency	sswld	pgeSswld
DpPrPgeExitMessage	sswld	pgeSswld
PIPgeMaster	sswld	sswld
PIDataProcessingRequest	sswld	sswld

Dependencies on Table: PIRscComputer

Reference by List

Referenced by	Primary Key	Foreign Key
DpPrRscCpuRamAllocation	computerId	computerId
PIRscDiskPartition	computerId	computerId

Dependencies on Table: PIRscDiskPartition

Reference by List

Referenced by	Primary Key	Foreign Key
DpPrDiskAllocation	diskPartitionId	diskPartitionId

Dependencies on Table: PIRscRealComputer

Reference by List

Referenced by	Primary Key	Foreign Key
PIRscComputer	realComputerId	realComputerId

Dependencies on Table: PIRscString

Reference by List

Referenced by	Primary Key	Foreign Key
PIRscComputer	stringId	stringId

Dependencies on Table: PI Tile Coordinates

Reference by List

Referenced by	Primary Key	Foreign Key
PIDataGranuleShort	tileId	tileId
PIDataProcessingRequest	tileId	tileId
PI Tile Schema Short	tileId	tileId
PI Tile Schema Short	tileId	tileId
PIDataProcessingRequest	tileId	tileId
PIDataGranuleShort	tileId	tileId

Dependencies on Table: PI Tile Schema Short

Reference by List

Referenced by	Primary Key	Foreign Key
PIDataProcessingRequest	tileId	tileId
PIDataGranuleShort	tileId	tileId
PIPgeDetailTile	schemaName	schemaName
PIPgeDetailTile	schemaName	schemaName

Dependencies on Table: Rp Activity Type

Reference by List

Referenced by	Primary Key	Foreign Key
RpResourceReservation	activityTypId	activityTypId

Dependencies on Table: Rp Resource Reservation

Reference by List

Referenced by	Primary Key	Foreign Key
RpRsvRscExplosion	reservationId	reservationId
RpReservationInterval	reservationId	reservationId

4.1.9 Triggers

Sybase supports the enforcement of business policy via the use of triggers. A trigger is best defined as set of activities or checks that should be performed automatically whenever a row is inserted, updated, or deleted from a given table. Sybase allows the definition of insert, update, and delete trigger per table. A listing of each the triggers in the PDPS database is given here.

Trigger List

Table	Trigger	User-Defined
DpPrCreationQueue	TrigUpdDpPrCreationQueue	Yes
DpPrCreationQueue	TrigDelDpPrCreationQueue	Yes
DpPrDiskAllocation	TrigInsDpPrDiskAllocation	Yes
DpPrDiskAllocation	TrigUpdDpPrDiskAllocation	Yes
DpPrDprDependList	TrigInsDpPrDprDependList	Yes
DpPrDprDependList	TrigUpdDpPrDprDependList	Yes
DpPrExecutable	TrigUpdDpPrExecutable	Yes
DpPrExecutable	TrigDelDpPrExecutable	Yes
DpPrFile	TrigUpdDpPrFile	Yes
DpPrPcf	TrigInsDpPrPcf	Yes
DpPrPcf	TrigUpdDpPrPcf	Yes
DpPrPge	TrigInsDpPrPge	Yes
DpPrPge	TrigUpdDpPrPge	Yes
DpPrPge	TrigDelDpPrPge	Yes
PIAlternate	TrigInsPIAlternate	Yes
PIAlternate	TrigUpdPIAlternate	Yes
PIAlternateInputValues	TrigInsPIAlternateInputValues	Yes
PIAlternateInputValues	TrigUpdPIAlternateInputValues	Yes
PIAssociatedScienceData	TrigInsPIAssociatedScienceData	Yes
PIAssociatedScienceData	TrigUpdPIAssociatedScienceData	Yes
PIDataGranuleShort	TrigInsPIDataGranuleShort	Yes
PIDataGranuleShort	TrigUpdPIDataGranuleShort	Yes
PIDataGranuleShort	TrigDelPIDataGranuleShort	Yes
PIDataProcessingRequest	TrigInsPIDataProcessingRequest	Yes
PIDataProcessingRequest	TrigUpdPIDataProcessingRequest	Yes
PIDataProcessingRequest	TrigDelPIDataProcessingRequest	Yes
PIDataSourceMaster	TrigInsPIDataSourceMaster	Yes
PIDataSourceMaster	TrigUpdPIDataSourceMaster	Yes
PIDataSourceMaster	TrigDelPIDataSourceMaster	Yes
PIDataTypeMaster	TrigUpdPIDataTypeMaster	Yes
PIDataTypeMaster	TrigDelPIDataTypeMaster	Yes
PIDataTypeReq	TrigInsPIDataTypeReq	Yes
PIDataTypeReq	TrigUpdPIDataTypeReq	Yes
PIDataTypeReq	TrigDelPIDataTypeReq	Yes
PIDprData	TrigInsPIDprData	Yes
PIDprData	TrigUpdPIDprData	Yes
PIDprData	TrigDelPIDprData	Yes
PIEsdtParam	TrigInsPIEsdtParam	Yes
PIEsdtParam	TrigUpdPIEsdtParam	Yes
PIEsdtParamValues	TrigInsPIEsdtParamValues	Yes

Table	Trigger	User-Defined
PIEsdtParmValues	TrigUpdPIEsdtParmValues	Yes
PIMetadataChecks	TrigInsPIMetadataChecks	Yes
PIMetadataChecks	TrigUpdPIMetadataChecks	Yes
PIOOutputDataYield	TrigInsPIOOutputDataYield	Yes
PIOOutputDataYield	TrigUpdPIOOutputDataYield	Yes
PIOOutputDataYield	TrigDelPIOOutputDataYield	Yes
PIPgeDetailParameters	TrigInsPIPgeDetailParameters	Yes
PIPgeDetailParameters	TrigUpdPIPgeDetailParameters	Yes
PIPgeDetailParameters	TrigDelPIPgeDetailParameters	Yes
PIPgeDetailTile	TrigInsPIPgeDetailTile	Yes
PIPgeDetailTile	TrigUpdPIPgeDetailTile	Yes
PIPgeDetailTime	TrigInsPIPgeDetailTime	Yes
PIPgeDetailTime	TrigUpdPIPgeDetailTime	Yes
PIPgeMaster	TrigUpdPIPgeMaster	Yes
PIPgeMaster	TrigDelPIPgeMaster	Yes
PIPgePerformance	TrigInsPIPgePerformance	Yes
PIPgePerformance	TrigUpdPIPgePerformance	Yes
PIPgePriority	TrigInsPIPgePriority	Yes
PIPgePriority	TrigUpdPIPgePriority	Yes
PIPlans	TrigInsPIPlans	Yes
PIPlans	TrigUpdPIPlans	Yes
PIPlans	TrigDelPIPlans	Yes
PIPrMetadataValue	TrigInsPIPrMetadataValue	Yes
PIPrMetadataValue	TrigUpdPIPrMetadataValue	Yes
PIProductionRequest	TrigInsPIProductionRequest	Yes
PIProductionRequest	TrigUpdPIProductionRequest	Yes
PIProductionRequest	TrigDelPIProductionRequest	Yes
PIProductionStrategy	TrigUpdPIProductionStrategy	Yes
PIProductionStrategy	TrigDelPIProductionStrategy	Yes
PIPrParameter	TrigInsPIPrParameter	Yes
PIPrParameter	TrigUpdPIPrParameter	Yes
PIPRPriority	TrigInsPIPRPriority	Yes
PIPRPriority	TrigUpdPIPRPriority	Yes
PIPrVsPlan	TrigInsPIPrVsPlan	Yes
PIPrVsPlan	TrigUpdPIPrVsPlan	Yes
PIRescUseThresh	TrigInsPIRescUseThresh	Yes
PIRescUseThresh	TrigInsPIRescUseThresh	Yes
PIResource	TrigUpdPIResource	Yes
PIResource	TrigDelPIResource	Yes
PIResourceRequirement	TrigInsPIResourceRequirement	Yes
PIResourceRequirement	TrigUpdPIResourceRequirement	Yes

Table	Trigger	User-Defined
PIResourceRequirement	TrigDelPIResourceRequirement	Yes
PIRoutineArrival	TrigInsPIRoutineArrival	Yes
PIRoutineArrival	TrigUpdPIRoutineArrival	Yes
PIRscComputer	TrigInsPIRscComputer	Yes
PIRscComputer	TrigUpdPIRscComputer	Yes
PIRscComputer	TrigDelPIRscComputer	Yes
PIRscDiskPartition	TrigInsPIRscDiskPartition	Yes
PIRscDiskPartition	TrigUpdPIRscDiskPartition	Yes
PIRscDiskPartition	TrigDelPIRscDiskPartition	Yes
PIRscString	TrigInsPIRscString	Yes
PIRscString	TrigUpdPIRscString	Yes
PIRscString	TrigDelPIRscString	Yes
PITileCoordinates	TrigUpdPITileCoordinates	Yes
PITileCoordinates	TrigDelPITileCoordinates	Yes
PITileSchemaShort	TrigInsPITileSchemaShort	Yes
PITileSchemaShort	TrigUpdPITileSchemaShort	Yes
PITileSchemaShort	TrigDelPITileSchemaShort	Yes
PIUsedByCenter	TrigInsPIUsedByCenter	Yes
PIUsedByCenter	TrigUpdPIUsedByCenter	Yes
PIUserPriority	TrigInsPIUserPriority	Yes
PIUserPriority	TrigUpdPIUserPriority	Yes

Trigger: TrigUpdDpPrCreationQueue

Trigger Code

```

CREATE TRIGGER TrigUpdDpPrCreationQueue
ON DpPrCreationQueue
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int /* number of rows updated */
SELECT @numrows = @@rowcount

/* if no rows were deleted, then return */

IF @numrows = 0
    RETURN

/* see if the primary key was updated */

```

```

IF UPDATE ( dprId )
BEGIN

    /* update of pk on multiple rows not allowed */

    IF @numrows > 1
    BEGIN
        RAISERROR 81113 "Update to primary key on multiple rows is not supported"
        ROLLBACK TRANSACTION
        RETURN
    END

    /* cascade update to child table */

    UPDATE DpPrDprDependList
    SET t.dprId = i.dprId
    FROM DpPrDprDependList t, inserted i, deleted d
    WHERE t.dprId = d.dprId

    IF @@transtate = 2 /* previous stmt aborted */
    BEGIN
        ROLLBACK TRANSACTION
        RETURN
    END

    END

    RETURN

END
go

```

Trigger: TrigDelDpPrCreationQueue

Trigger Code

```

CREATE TRIGGER TrigDelDpPrCreationQueue
ON DpPrCreationQueue
FOR DELETE
AS
BEGIN

```

```

/* if no rows were deleted, then return */

IF @@rowcount = 0
    RETURN

/* delete matching rows in table DpPrDprDependList */

DELETE DpPrDprDependList FROM DpPrDprDependList t, deleted d
WHERE t.dprId = d.dprId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

RETURN

END
go

```

Trigger: TrigInsDpPrDiskAllocation

Trigger Code

```

CREATE TRIGGER TrigInsDpPrDiskAllocation
ON DpPrDiskAllocation
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key diskPartitionId has a
   matching primary key in table PlRscDiskPartition */

IF (SELECT COUNT(*)
    FROM PlRscDiskPartition t, inserted i
    WHERE t.diskPartitionId = i.diskPartitionId
    AND t.computerId = i.computerId)

```

```

!= @numrows
BEGIN
    RAISERROR 81111 "No match for (diskPartitionId,computerId) found in table
PIRscDiskPartition"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdDpPrDiskAllocation

Trigger Code

```

CREATE TRIGGER TrigUpdDpPrDiskAllocation
ON DpPrDiskAllocation
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

IF UPDATE ( diskPartitionId ) OR UPDATE ( computerId )
BEGIN

/* Verify that foreign key diskPartitionId has a
 * matching primary key in table PIRscDiskPartition */

IF (SELECT COUNT(*)
    FROM PIRscDiskPartition t, inserted i
    WHERE t.diskPartitionId = i.diskPartitionId
    AND t.computerId = i.computerId)
!= @numrows
BEGIN
    RAISERROR 81112 "No match for (diskPartitionId,computerId) found in table

```

```

PIRscDiskPartition"
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN

END
go

```

Trigger: TrigInsDpPrDprDependList

Trigger Code

```

CREATE TRIGGER TrigInsDpPrDprDependList
ON DpPrDprDependList
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key dprId has a
 * matching primary key in table DpPrCreationQueue */

IF (SELECT COUNT(*)
    FROM DpPrCreationQueue t, inserted i
    WHERE t.dprId = i.dprId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for dprId found in table DpPrCreationQueue"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

```

```
END  
go
```

Trigger: TrigUpdDpPrDprDependList

Trigger Code

```
CREATE TRIGGER TrigUpdDpPrDprDependList  
ON DpPrDprDependList  
FOR UPDATE  
AS  
BEGIN  
  
/* See how many rows were updated */  
  
DECLARE @numrows int  
SELECT @numrows = @@rowcount  
IF @numrows = 0  
    RETURN  
  
IF UPDATE ( dprId )  
BEGIN  
  
    /* Verify that foreign key dprId has a  
     * matching primary key in table DpPrCreationQueue */  
  
    IF (SELECT COUNT(*)  
        FROM DpPrCreationQueue t, inserted i  
        WHERE t.dprId = i.dprId)  
        != @numrows  
    BEGIN  
        RAISERROR 81112 "No match for dprId found in table DpPrCreationQueue"  
        ROLLBACK TRANSACTION  
        RETURN  
    END  
  
    END  
  
    RETURN  
  
END  
  
go
```

Trigger: TrigUpdDpPrExecutable

Trigger Code

```
CREATE TRIGGER TrigUpdDpPrExecutable
ON DpPrExecutable
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int /* number of rows updated */
SELECT @numrows = @@rowcount

/* if no rows were deleted, then return */

IF @numrows = 0
    RETURN

/* see if the primary key was updated */

IF UPDATE(sswId) OR UPDATE(execName) OR UPDATE(execMachine)
BEGIN

/* update of pk on multiple rows not allowed */

IF @numrows > 1
BEGIN
    RAISERROR 81113 "Update to primary key on multiple rows is not supported"
    ROLLBACK TRANSACTION
    RETURN
END

/* cascade update to child table */

UPDATE DpPrPge
SET t.sswId = i.sswId,
    t.execName = i.execName,
    t.execMachine = i.execMachine
FROM DpPrPge t, inserted i, deleted d
WHERE t.sswId = d.sswId AND
```

```

t.execName = d.execName AND
t.execMachine = d.execMachine

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN

END
go

```

Trigger: TrigDelDpPrExecutable

Trigger Code

```

CREATE TRIGGER TrigDelDpPrExecutable
ON DpPrExecutable
FOR DELETE
AS
BEGIN

/* if no rows were deleted, then return */

IF @@rowcount = 0
    RETURN

/* delete matching rows in table DpPrPge */

DELETE DpPrPge FROM DpPrPge t, deleted d
WHERE t.sswId = d.sswId
AND t.execName = d.execName
AND t.execMachine = d.execMachine

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN

```

```
END  
RETURN  
END  
go
```

Trigger: TrigInsDpPrPcf

Trigger Code

```
CREATE TRIGGER TrigUpdDpPrFile  
ON DpPrFile  
FOR UPDATE  
AS  
  
DECLARE @numrows int /* number of rows updated */  
  
BEGIN  
  
SELECT @numrows = @@rowcount  
  
/* if no rows were deleted, then return */  
  
IF @numrows = 0  
    RETURN  
  
/* see if the filename was updated */  
  
IF UPDATE ( fileName )  
BEGIN  
  
/* cascade update to table DpPrDiskAllocation */
```

```

UPDATE DpPrDiskAllocation
SET t.fileName = i.fileName
FROM DpPrDiskAllocation t,
     inserted i, deleted d
WHERE t.fileName = d.fileName
AND t.diskPartitionId = d.diskPartitionId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

END
RETURN
END
go

```

Trigger: TrigInsDpPrPcf

Trigger Code

```

CREATE TRIGGER TrigInsDpPrPcf
ON DpPrPcf
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

```

```

/* Verify that foreign key has a matching
 * primary key in table DpPrPge      */

IF (SELECT COUNT(*)
    FROM DpPrPge t, inserted i
    WHERE t.dprId = i.dprId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for dprId in table DpPrPge"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdDpPrPcf

Trigger Code

```

CREATE TRIGGER TrigUpdDpPrPcf
ON DpPrPcf
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

IF UPDATE ( dprId )
BEGIN

/* Verify that foreign key has a matching
 * primary key in table DpPrPge      */

IF (SELECT COUNT(*)
    FROM DpPrPge t, inserted i
    WHERE t.dprId = i.dprId)
!= @numrows

```

```

BEGIN
    RAISERROR 81112 "No match for dprId in table DpPrPge"
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN

END
go

```

Trigger: TrigInsDpPrPge

Trigger Code

```

CREATE TRIGGER TrigInsDpPrPge
ON DpPrPge
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key has a matching
 * primary key in table DpPrExecutable      */

IF (SELECT COUNT(*)
    FROM DpPrExecutable t, inserted i
    WHERE t.sswId = i.sswId
    AND  t.execName = i.execName
    AND  t.execMachine = i.execMachine)
    != @numrows
BEGIN
    RAISERROR 81111 "No match for (sswId/execName/execMachine) in table
DpPrExecutable"
    ROLLBACK TRANSACTION
    RETURN

```

```
END  
RETURN  
END  
go
```

Trigger: TrigUpdDpPrPge

Trigger Code

```
CREATE TRIGGER TrigUpdDpPrPge  
ON DpPrPge  
FOR UPDATE  
AS  
BEGIN  
  
/* See how many rows were updated */  
  
DECLARE @numrows int /* number of rows updated */  
SELECT @numrows = @@rowcount  
  
/* if no rows were updated, then return */  
  
IF @numrows = 0  
    RETURN  
  
/* see if the primary key was updated */  
  
IF UPDATE ( dprId )  
BEGIN  
  
/* update of pk on multiple rows not allowed */  
  
IF @numrows > 1  
BEGIN  
    RAISERROR 81113 "Update to primary key on multiple rows is not supported"  
    ROLLBACK TRANSACTION  
    RETURN  
END  
  
/* cascade update to child table */
```

```

UPDATE DpPrPcf
SET t.dprId = i.dprId
FROM DpPrPcf t, inserted i, deleted d
WHERE t.dprId = d.dprId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

/* see if any foreign keys were updated */

IF UPDATE ( sswId ) OR UPDATE ( execName ) OR UPDATE ( execMachine )
BEGIN
    IF (SELECT COUNT(*)
        FROM DpPrExecutable t, inserted i
        WHERE t.sswId = i.sswId
        AND t.execName = i.execName
        AND t.execMachine = i.execMachine)
        != @numrows
    BEGIN
        RAISERROR 81112 "No match for (sswId/execName/execMachine) in table
DpPrExecutable"
        ROLLBACK TRANSACTION
        RETURN
    END
END

RETURN

END
go

```

Trigger: TrigDelDpPrPge

Trigger Code

```

CREATE TRIGGER TrigDelDpPrPge
ON DpPrPge
FOR DELETE

```

```

AS
BEGIN

/* if no rows were deleted, then return */

IF @@rowcount = 0
    RETURN

/* delete matching rows in table DpPrPcf */

DELETE DpPrPcf FROM DpPrPcf t, deleted d
WHERE t.dprId = d.dprId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigInsPlAlternate

Trigger Code

```

CREATE TRIGGER TrigInsPlAlternate
ON PlAlternate
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that key (pgId, dataTypeId, logicalId) has a matching
primary key in table PlDataTypeReq */

```

```

IF (SELECT COUNT(*)
    FROM PlDataTypeReq t, inserted i
    WHERE t.pgeId = i.pgeId
    AND t.dataTypeId = i.dataTypeId
    AND t.logicalId = i.logicalId) != @numrows
BEGIN
    RAISERROR 81111 "No match for (pgeId, dataTypeId, logicalId) in PlDataTypeReq"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPlAlternate

Trigger Code

```

CREATE TRIGGER TrigUpdPlAlternate
ON PlAlternate
FOR UPDATE
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that key (pgeId, dataTypeId, logicalId) has a matching
   primary key in table PlDataTypeReq */


IF (SELECT COUNT(*)
    FROM PlDataTypeReq t, inserted i
    WHERE t.pgeId = i.pgeId
    AND t.dataTypeId = i.dataTypeId
    AND t.logicalId = i.logicalId) != @numrows
BEGIN
    RAISERROR 81112 "Invalid foreign key (no matching primary key in PlDataTypeReq"

```

```
ROLLBACK TRANSACTION  
RETURN  
END  
  
RETURN  
  
END  
go
```

Trigger: TrigInsPlAlternateInputValues

Trigger Code

```
CREATE TRIGGER TrigInsPlAlternateInputValues  
ON PlAlternateInputValues  
FOR INSERT  
AS  
BEGIN  
  
/* See how many rows were inserted */  
  
DECLARE @numrows int  
SELECT @numrows = @@rowcount  
IF @numrows = 0  
    RETURN  
  
/* Verify that every foreign key has a matching  
primary key in table PlProductionRequest */  
  
IF (SELECT COUNT(*)  
    FROM PlProductionRequest t, inserted i  
    WHERE t.productionRequestId = i.productionRequestId)  
!= @numrows  
BEGIN  
    RAISERROR 81111 "No match for productionRequestId in table PlProductionRequest"  
    ROLLBACK TRANSACTION  
    RETURN  
END  
  
RETURN  
  
END  
go
```

Trigger: TrigUpdPlAlternateInputValues

Trigger Code

```
CREATE TRIGGER TrigUpdPlAlternateInputValues
ON PlAlternateInputValues
FOR UPDATE
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that every foreign key has a matching
primary key in table PlProductionRequest */

IF (SELECT COUNT(*)
    FROM PlProductionRequest t, inserted i
    WHERE t.productionRequestId = i.productionRequestId)
!= @numrows
BEGIN
    RAISERROR 81112 "No match for productionRequestId in table PlProductionRequest"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go
```

Trigger: TrigInsPlAssociatedScienceData

Trigger Code

```
CREATE TRIGGER TrigInsPlAssociatedScienceData
ON PlAssociatedScienceData
FOR INSERT
AS
```

```

BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF numrows = 0
    RETURN
/* Verify that key (pgeId, dataTypeId, logicalId) has a matching
Primary key in the table PlOutputDataYield */
IF (SELECT COUNT(*)
    FROM PlOutputDataYield t, inserted i
    WHERE t.pgeId = i.pgeId
    AND t.dataTypeId - i.dataTypeId
    AND t.logicalId - i.logicalId) != @numrows
BEGIN
    RAISERROR 81111 "No match for (pgeId, dataTypeId, logicalId) in PlOutput"
    ROLLBACK TRANSACTION
    RETURN
END
RETURN
END
go

```

Trigger: TrigUpdPlAssociatedScienceData

Trigger Code

```

CREATE TRIGGER TrigUpdPlAssociatedScienceData
ON PlAssociatedScienceData
FOR UPDATE
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int

```

```

SELECT @numrows = @@rowcount
IF numrows = 0
    RETURN
/* Verify that key (pgeId, dataTypeId, logicalId) has a matching
Primary key in the table PIOutputDataYield */
IF (SELECT COUNT(*)
    FROM PIOutputDataYield t, inserted i
    WHERE t.pgeId = i.pgeId
    AND t.dataTypeId = i.dataTypeId
    AND t.logicalId = i.logicalId) != @numrows
BEGIN
    RAISERROR 81112 "Invalid foreign key (no matching key in PIOutput)"
    ROLLBACK TRANSACTION
    RETURN
END
RETURN
END
go

```

Trigger: TrigInsPlDataGranuleShort

Trigger Code

```

CREATE TRIGGER TrigInsPlDataGranuleShort
ON PlDataGranuleShort
FOR INSERT
AS
BEGIN
    /* See how many rows were inserted */

    DECLARE @numrows int
    SELECT @numrows = @@rowcount
    IF @numrows = 0
        RETURN

```

```

/* Verify that key tileId, if not null or zero, has a matching
 primary key in table PItileCoordinates */

IF (SELECT COUNT(*)
    FROM PItileCoordinates t, inserted i
    WHERE t.tileId = i.tileId) +
    (SELECT COUNT(*) FROM inserted i
    WHERE i.tileId = 0 ) +
    (SELECT COUNT(*) FROM inserted i
    WHERE i.tileId is null ) != @numrows
BEGIN
    RAISERROR 81111 "No match for tileId in table PItileCoordinates"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that key dataTypeId has a matching
 primary key in table PIDataTypeMaster */

IF (SELECT COUNT(*)
    FROM PIDataTypeMaster t, inserted i
    WHERE t.dataTypeId = i.dataTypeId) != @numrows
BEGIN
    RAISERROR 81111 "No match for dataTypeId in table PIDataTypeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPIDataGranuleShort

Trigger Code

```

CREATE TRIGGER TrigUpdPIDataGranuleShort
ON PIDataGranuleShort
FOR UPDATE
AS

```

```

DECLARE @numrows int /* number of rows updated */

BEGIN

SELECT @numrows = @@rowcount

/* if no rows were deleted, then return */

IF @numrows = 0
    RETURN

/* see if the primary key was updated */

IF UPDATE ( granuleId )
BEGIN

/* update of pk on multiple rows not allowed */

IF @numrows > 1
BEGIN
    RAISERROR 81113 "Update to primary key on multiple rows is not supported"
    ROLLBACK TRANSACTION
    RETURN
END

/* cascade update to child tables */

UPDATE PlDprData
SET t.granuleId = i.granuleId
FROM PlDprData t,
     inserted i, deleted d
WHERE t.granuleId = d.granuleId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PlEsdtParmValues
SET t.granuleId = i.granuleId
FROM PlEsdtParmValues t,
     inserted i, deleted d

```

```

WHERE t.granuleId = d.granuleId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

END

/* If foreign keys have been updated, verify that they
match the appropriate primary key. */

IF UPDATE ( tileId )
BEGIN

    /* Verify that key tileId, if not zero, has a
       matching primary key in table PItileCoordinates */

    IF (SELECT COUNT(*)
        FROM PItileCoordinates t, inserted i
        WHERE t.tileId = i.tileId) +
    (SELECT COUNT(*)
        FROM inserted i
        WHERE i.tileId = 0)
    != @numrows
    BEGIN
        RAISERROR 81112 "No match for tileId in table PItileCoordinates"
        ROLLBACK TRANSACTION
        RETURN
    END

    END

    IF UPDATE ( dataTypeId )
    BEGIN

        /* Verify that key dataTypeId has a matching
           primary key in table PIDataTypeMaster */

        IF (SELECT COUNT(*)
            FROM PIDataTypeMaster t, inserted i
            WHERE t.dataTypeId = i.dataTypeId) != @numrows
        BEGIN

```

```

RAISERROR 81112 "No match for dataTypeId in table PIDataTypeMaster"
ROLLBACK TRANSACTION
RETURN
END

END

IF UPDATE ( universalReference )
BEGIN

/* Cascade update to DpPrFile */

UPDATE DpPrFile
SET t.universalReference = i.universalReference
FROM DpPrFile t, inserted i, deleted d
WHERE t.universalReference = d.universalReference

END

-- Update the time stamp:

UPDATE PIDataGranuleShort
SET t.timeStamp = getdate()
FROM PIDataGranuleShort t, inserted i
WHERE t.granuleId = i.granuleId

RETURN

END
go

```

Trigger: TrigDelPIDataGranuleShort

Trigger Code

```

CREATE TRIGGER TrigDelPIDataGranuleShort
ON PIDataGranuleShort
FOR DELETE
AS
BEGIN

/* if no rows were deleted, then return */

```

```

IF @@rowcount = 0
    RETURN

/* delete matching rows in PlDprData */

DELETE PlDprData
FROM PlDprData t, deleted d
WHERE t.granuleId = d.granuleId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlEsdtParmValues */

DELETE PlEsdtParmValues
FROM PlEsdtParmValues t, deleted d
WHERE t.granuleId = d.granuleId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

RETURN

END
go

```

Trigger: TrigInsPlDataProcessingRequest

Trigger Code

```

CREATE TRIGGER TrigInsPlDataProcessingRequest
ON PlDataProcessingRequest
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

```

```

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key productionRequestId has a matching
   primary key in table PIProductionRequest */

IF (SELECT COUNT(*)
    FROM PIProductionRequest t, inserted i
    WHERE t.productionRequestId = i.productionRequestId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for productionRequestId in table PIProductionRequest"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that foreign key pgeId has a matching
   primary key in table PIPgeMaster */

IF (SELECT COUNT(*)
    FROM PIPgeMaster t, inserted i
    WHERE t.pgeId = i.pgeId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for pgeId in table PIPgeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that foreign key sswId, if specified, has a matching
   primary key in table PIResourceRequirement */

IF ((SELECT COUNT(*)
      FROM PIResourceRequirement t, inserted i
      WHERE t.sswId = i.sswId) +
    (SELECT COUNT(*)
      FROM inserted i
      WHERE i.sswId is null) +
    (SELECT COUNT(*)
      FROM inserted i
      WHERE i.sswId = ""))

```

```

!= @numrows
BEGIN
    RAISERROR 81111 "No match for sswId in table PIResourceRequirement"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that foreign key tileId, if specified, has a matching
   primary key in table PItileCoordinates */

IF ((SELECT COUNT(*)
      FROM PItileCoordinates t, inserted i
      WHERE t.tileId = i.tileId) +
    (SELECT COUNT(*)
      FROM inserted i
      WHERE i.tileId is null) +
    (SELECT COUNT(*)
      FROM inserted i
      WHERE i.tileId = 0)) != @numrows
BEGIN
    RAISERROR 81111 "No match for tileId in table PItileCoordinates"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPIDataProcessingRequest

Trigger Code

```

CREATE TRIGGER TrigUpdPIDataProcessingRequest
ON PIDataProcessingRequest
FOR UPDATE
AS

```

```

DECLARE @numrows int /* number of rows updated */

BEGIN

```

```

SELECT @numrows = @@rowcount

/* if no rows were deleted, then return */

IF @numrows = 0
    RETURN

/* see if the primary key was updated */

IF UPDATE ( dprId )
BEGIN

    /* update of pk on multiple rows not allowed */

    IF @numrows > 1
    BEGIN
        RAISERROR 81113 "Update to primary key on multiple rows is not supported"
        ROLLBACK TRANSACTION
        RETURN
    END

    /* cascade update to child tables */

    UPDATE PlDprData
    SET t.dprId = i.dprId
    FROM PlDprData t,
         inserted i, deleted d
    WHERE t.dprId = d.dprId

    IF @@transtate = 2 /* previous stmt aborted */
    BEGIN
        ROLLBACK TRANSACTION
        RETURN
    END

END

-- Update the time stamp:

UPDATE PlDataProcessingRequest
SET t.timeStamp = getdate()
FROM PlDataProcessingRequest t, inserted i
WHERE t.dprId = i.dprId

```

```
RETURN
```

```
END
```

```
go
```

Trigger: TrigDelPlDataProcessingRequest

Trigger Code

```
/* Add code for triggers */
```

```
CREATE TRIGGER TrigDelPlDataProcessingRequest
ON PlDataProcessingRequest
FOR DELETE
AS
BEGIN
```

```
/* if no rows were deleted, then return */
```

```
IF @@rowcount = 0
    RETURN
```

```
/* delete matching rows in PlDprData */
```

```
DELETE PlDprData
FROM PlDprData t, deleted d
WHERE t.dprId = d.dprId
```

```
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END
```

```
RETURN
```

```
END
```

```
go
```

Trigger: TrigInsPlDataSourceMaster

Trigger Code

```
CREATE TRIGGER TrigInsPlDataSourceMaster
ON PlDataSourceMaster
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that key dataTypeId has a matching
primary key in table PlDataTypeMaster */

IF (SELECT COUNT(*)
    FROM PlDataTypeMaster t, inserted i
    WHERE t.dataTypeId = i.dataTypeId) != @numrows
BEGIN
    RAISERROR 81111 "No match for dataTypeId found in table PlDataTypeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go
```

Trigger: TrigUpdPlDataSourceMaster

Trigger Code

```
CREATE TRIGGER TrigUpdPlDataSourceMaster
ON PlDataSourceMaster
FOR UPDATE
AS
BEGIN

DECLARE @numrows int /* number of rows updated */
SELECT @numrows = @@rowcount
```

```

/* if no rows were deleted, then return */

IF @numrows = 0
    RETURN

/* see if dataTypeId was updated */

IF UPDATE ( dataTypeId )
BEGIN

    /* update of pk on multiple rows not allowed */

    IF @numrows > 1
    BEGIN
        RAISERROR 81113 "Update to primary key on multiple rows is not supported"
        ROLLBACK TRANSACTION
        RETURN
    END
    /* This is a foreign key, as well as primary */
    /* First verify that the new key is valid */

    IF (SELECT COUNT(*)
        FROM PIDataTypeMaster t, inserted i
        WHERE t.dataTypeId = i.dataTypeId) != @numrows
    BEGIN
        RAISERROR 81112 "No match for dataTypeId found in table PIDataTypeMaster"
        ROLLBACK TRANSACTION
        RETURN
    END
    /* cascade update to child table */

    UPDATE PIRoutineArrival
    SET t.dataTypeId = i.dataTypeId
    FROM PIRoutineArrival t,
         inserted i, deleted d
    WHERE t.dataTypeId = d.dataTypeId

    IF @@transtate = 2 /* previous stmt aborted */
    BEGIN
        ROLLBACK TRANSACTION
        RETURN
    END

```

```
END  
  
RETURN  
  
END  
go
```

Trigger: TrigDelPlDataSourceMaster

Trigger Code

```
CREATE TRIGGER TrigDelPlDataSourceMaster  
ON PlDataSourceMaster  
FOR DELETE  
AS  
BEGIN  
  
/* if no rows were deleted, then return */  
  
IF @@rowcount = 0  
    RETURN  
  
/* delete matching rows in PlRoutineArrival */  
  
DELETE PlRoutineArrival  
FROM PlRoutineArrival t, deleted d  
WHERE t.dataTypeId = d.dataTypeId  
  
IF @@transtate = 2 /* previous statement aborted */  
BEGIN  
    ROLLBACK TRANSACTION /* undo all deletes */  
    RETURN  
END  
  
RETURN  
  
END  
go
```

Trigger: TrigUpdPlDataTypeMaster

Trigger Code

```
CREATE TRIGGER TrigUpdPlDataTypeMaster
ON PlDataTypeMaster
FOR UPDATE
AS

DECLARE @numrows int /* number of rows updated */

BEGIN

SELECT @numrows = @@rowcount

/* if no rows were deleted, then return */

IF @numrows = 0
    RETURN

/* see if the primary key was updated */

IF UPDATE ( dataTypeId )
BEGIN

/* update of pk on multiple rows not allowed */

IF @numrows > 1
BEGIN
    RAISERROR 81113 "Update to primary key on multiple rows is not supported"
    ROLLBACK TRANSACTION
    RETURN
END

/* cascade update to child tables */

UPDATE PlDataGranuleShort
SET t.dataTypeId = i.dataTypeId
FROM PlDataGranuleShort t,
     inserted i, deleted d
WHERE t.dataTypeId = d.dataTypeId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
```

```

        RETURN
END

UPDATE PIDataSourceMaster
SET  t.dataTypeId = i.dataTypeId
FROM  PIDataSourceMaster t,
      inserted i, deleted d
WHERE  t.dataTypeId = d.dataTypeId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PIEsdtParam
SET  t.dataTypeId = i.dataTypeId
FROM  PIEsdtParam t,
      inserted i, deleted d
WHERE  t.dataTypeId = d.dataTypeId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PIUsedByCenter
SET  t.dataTypeId = i.dataTypeId
FROM  PIUsedByCenter t,
      inserted i, deleted d
WHERE  t.dataTypeId = d.dataTypeId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PIDataTypeReq
SET  t.dataTypeId = i.dataTypeId
FROM  PIDataTypeReq t,
      inserted i, deleted d
WHERE  t.dataTypeId = d.dataTypeId

```

```

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PIOutputDataYield
SET t.dataTypeId = i.dataTypeId
FROM PIOutputDataYield t,
     inserted i, deleted d
WHERE t.dataTypeId = d.dataTypeId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN

END
go

```

Trigger: TrigDelPlDataTypeMaster

Trigger Code

```

CREATE TRIGGER TrigDelPlDataTypeMaster
ON PlDataTypeMaster
FOR DELETE
AS
BEGIN

/* if no rows were deleted, then return */

IF @@rowcount = 0
    RETURN

/* delete matching rows in PlDataSourceMaster */

```

```

DELETE PIDataSourceMaster
FROM PIDataSourceMaster t, deleted d
WHERE t.dataTypeId = d.dataTypeId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PIEsdtParam */

DELETE PIEsdtParam
FROM PIEsdtParam t, deleted d
WHERE t.dataTypeId = d.dataTypeId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PIUsedByCenter */

DELETE PIUsedByCenter
FROM PIUsedByCenter t, deleted d
WHERE t.dataTypeId = d.dataTypeId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PIDataTypeReq */

DELETE PIDataTypeReq
FROM PIDataTypeReq t, deleted d
WHERE t.dataTypeId = d.dataTypeId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN

```

```

END

/* delete matching rows in PIOutputDataYield */

DELETE PIOutputDataYield
FROM PIOutputDataYield t, deleted d
WHERE t.dataTypeId = d.dataTypeId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

RETURN

END
go

```

Trigger: TrigInsPlDataTypeReq

Trigger Code

```

CREATE TRIGGER TrigInsPlDataTypeReq
ON PlDataTypeReq
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key pgeId has a matching
primary key in table PlPgeMaster */

IF (SELECT COUNT(*)
    FROM PlPgeMaster t, inserted i
    WHERE t.pgeId = i.pgeId)
!= @numrows

```

```

BEGIN
    RAISERROR 81111 "No match for pgeId in table PlPgeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that foreign key dataTypeId has a matching
   primary key in table PlDataTypeMaster      */

IF (SELECT COUNT(*)
    FROM PlDataTypeMaster t, inserted i
    WHERE t.dataTypeId = i.dataTypeId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for dataTypeId in table PlDataTypeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPlDataTypeReq

Trigger Code

```

CREATE TRIGGER TrigUpdPlDataTypeReq
ON PlDataTypeReq
FOR UPDATE
AS

DECLARE @numrows int /* number of rows updated */

BEGIN

SELECT @numrows = @@rowcount

/* if no rows were deleted, then return */

IF @numrows = 0
    RETURN

```

```

/* see if any foreign keys were updated */

IF UPDATE ( pgeId )
BEGIN

    /* Verify that foreign key pgeId has a matching
     * primary key in table PlPgeMaster          */

    IF (SELECT COUNT(*)
        FROM PlPgeMaster t, inserted i
        WHERE t.pgeId = i.pgeId)
        != @numrows
    BEGIN
        RAISERROR 81112 "No match for pgeId in table PlPgeMaster"
        ROLLBACK TRANSACTION
        RETURN
    END
END

IF UPDATE ( dataTypeId )
BEGIN

    /* Verify that foreign key dataTypeId has a matching
     * primary key in table PlDataTypeMaster      */

    IF (SELECT COUNT(*)
        FROM PlDataTypeMaster t, inserted i
        WHERE t.dataTypeId = i.dataTypeId)
        != @numrows
    BEGIN
        RAISERROR 81112 "No match for dataTypeId in table PlDataTypeMaster"
        ROLLBACK TRANSACTION
        RETURN
    END
END

/* see if the primary key was updated */

IF UPDATE ( pgeId ) OR UPDATE ( dataTypeId ) OR UPDATE ( logicalId )
BEGIN

```

```

/* update of pk on multiple rows not allowed */

IF @numrows > 1
BEGIN
    RAISERROR 81113 "Update to primary key on multiple rows is not supported"
    ROLLBACK TRANSACTION
    RETURN
END

/* cascade update to child tables */

UPDATE PlMetadataChecks
SET t.pgeId = i.pgeId,
    t.dataTypeId = i.dataTypeId,
    t.logicalId = i.logicalId
FROM PlMetadataChecks t,
     inserted i, deleted d
WHERE t.pgeId = d.pgeId
AND t.dataTypeId = d.dataTypeId
AND t.logicalId = d.logicalId
AND t.ioFlag = 0

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PlAlternate
SET t.pgeId = i.pgeId,
    t.dataTypeId = i.dataTypeId,
    t.logicalId = i.logicalId
FROM PlAlternate t,
     inserted i, deleted d
WHERE t.pgeId = d.pgeId
AND t.dataTypeId = d.dataTypeId
AND t.logicalId = d.logicalId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

```

```

UPDATE PlFileTypeReq
SET  t.pgeId = i.pgeId,
     t.dataTypeId = i.dataTypeId,
     t.logicalId = i.logicalId
FROM  PlFileTypeReq t,
      inserted i, deleted d
WHERE t.pgeId = d.pgeId
AND   t.dataTypeId = d.dataTypeId
AND   t.logicalId = d.logicalId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN

END
go

```

Trigger: TrigDelPlDataTypeReq

Trigger Code

```

CREATE TRIGGER TrigDelPlDataTypeReq
ON PlDataTypeReq
FOR DELETE
AS
BEGIN

/* if no rows were deleted, then return */

IF @@rowcount = 0
    RETURN

/* delete matching rows in PlMetadataChecks */

DELETE PlMetadataChecks
FROM  PlMetadataChecks t, deleted d
WHERE t.pgeId = d.pgeId

```

```

AND t.dataTypeId = d.dataTypeId
AND t.logicalId = d.logicalId
AND t.ioFlag = 0

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlAlternate */

DELETE PlAlternate
FROM PlAlternate t, deleted d
WHERE t.pgeId = d.pgeId
AND t.dataTypeId = d.dataTypeId
AND t.logicalId = d.logicalId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlFileTypeReq */

DELETE PlFileTypeReq
FROM PlFileTypeReq t, deleted d
WHERE t.pgeId = d.pgeId
AND t.dataTypeId = d.dataTypeId
AND t.logicalId = d.logicalId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

RETURN

END
go

```

Trigger: TrigInsPlDprData

Trigger Code

```
CREATE TRIGGER TrigInsPlDprData
ON PlDprData
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that key dprId has a matching
primary key in table PlDataProcessingRequest */

IF (SELECT COUNT(*)
    FROM PlDataProcessingRequest t, inserted i
    WHERE t.dprId = i.dprId) != @numrows
BEGIN
    RAISERROR 81111 "No match for dprId in table PlDataProcessingRequest"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that key granuleId has a matching
primary key in table PlDataGranuleShort */
```

*/

```
IF (SELECT COUNT(*)
    FROM PlDataGranuleShort t, inserted i
    WHERE t.granuleId = i.granuleId) != @numrows
BEGIN
    RAISERROR 81111 "No match for granuleId in table PlDataGranuleShort"
    ROLLBACK TRANSACTION
    RETURN
END
RETURN
END
go
```

Trigger: TrigUpdPlDprData

Trigger Code

```
CREATE TRIGGER TrigUpdPlDprData
ON PlDprData
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that key dprId has a matching
primary key in table PlDataProcessingRequest */

IF UPDATE ( dprId )
BEGIN
    IF (SELECT COUNT(*)
        FROM PlDataProcessingRequest t, inserted i
        WHERE t.dprId = i.dprId) != @numrows
    BEGIN
        RAISERROR 81112 "No match for dprId in table PlDataProcessingRequest"
        ROLLBACK TRANSACTION
        RETURN
    END
END
END

/* Verify that key granuleId has a matching
primary key in table PlDataGranuleShort */
```

/*

```
IF UPDATE ( granuleId )
BEGIN
    IF (SELECT COUNT(*)
        FROM PlDataGranuleShort t, inserted i
        WHERE t.granuleId = i.granuleId) != @numrows
    BEGIN
        RAISERROR 81112 "No match for granuleId in table PlDataGranuleShort"
```

```

ROLLBACK TRANSACTION
RETURN
END
END

RETURN

END
go

```

Trigger: TrigDelPlDprData

Trigger Code

```

CREATE TRIGGER TrigDelPlDprData
ON PlDprData
FOR DELETE
AS
BEGIN

/* if no rows were deleted, then return */

IF @@rowcount = 0
    RETURN

/* delete matching rows in PlTimer */

DELETE PlTimer
FROM PlTimer t, deleted d
WHERE t.dprId = d.dprId
AND t.granuleId = d.granuleId
AND t.logicalId = d.logicalId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

RETURN

END
go

```

Trigger: TrigInsPlEsdtParam

Trigger Code

```
CREATE TRIGGER TrigInsPlEsdtParam
ON PlEsdtParam
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that every foreign key has a matching
primary key in table PlDataTypeMaster      */

IF (SELECT COUNT(*)
    FROM PlDataTypeMaster t, inserted i
    WHERE t.dataTypeId = i.dataTypeId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for dataTypeId in table PlDataTypeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go
```

Trigger: TrigUpdPlEsdtParam

Trigger Code

```
CREATE TRIGGER TrigUpdPlEsdtParam
```

```

ON PlEsdtParam
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that every foreign key has a matching
primary key in table PlDataTypeMaster      */

IF UPDATE ( dataTypeId )
BEGIN
    IF (SELECT COUNT(*)
        FROM PlDataTypeMaster t, inserted i
        WHERE t.dataTypeId = i.dataTypeId)
        != @numrows
    BEGIN
        RAISERROR 81112 "No match for dataTypeId in table PlDataTypeMaster"
        ROLLBACK TRANSACTION
        RETURN
    END
    END
    RETURN

END
go

```

Trigger: TrigInsPlEsdtParmValues

Trigger Code

```

CREATE TRIGGER TrigInsPlEsdtParmValues
ON PlEsdtParmValues
FOR INSERT
AS
BEGIN

```

```

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that every foreign key has a matching
   primary key in table PlDataGranuleShort */

IF (SELECT COUNT(*)
    FROM PlDataGranuleShort t, inserted i
    WHERE t.granuleId = i.granuleId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for granuleId in table PlDataGranuleShort"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPlEsdtParmValues

Trigger Code

```

CREATE TRIGGER TrigUpdPlEsdtParmValues
ON PlEsdtParmValues
FOR UPDATE
AS
BEGIN

```

```

/* See how many rows were updated */

```

```

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

```

```

/* Verify that every foreign key has a matching

```

```

/* primary key in table PlDataGranuleShort */
```

```

IF UPDATE ( granuleId )
BEGIN
    IF (SELECT COUNT(*)
        FROM PlDataGranuleShort t, inserted i
        WHERE t.granuleId = i.granuleId)
        != @numrows
    BEGIN
        RAISERROR 81112 "No match for granuleId in table PlDataGranuleShort"
        ROLLBACK TRANSACTION
        RETURN
    END
END
RETURN
```

```

END
go
```

Trigger: TrigInsPlMetadataChecks

Trigger Code

```

CREATE TRIGGER TrigInsPlMetadataChecks
ON PlMetadataChecks
FOR INSERT
AS
BEGIN
```

```

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN
```

```

/* Verify that foreign key (pgeId, dataTypeId) has a matching
primary key in table PlDataTypeReq or PlOutputDataYield */

IF (SELECT COUNT(*)
    FROM PlDataTypeReq t, inserted i
    WHERE i.ioFlag = 0
```

```

        AND t.pgeId = i.pgeId
        AND t.logicalId = i.logicalId
        AND t.dataTypeId = i.dataTypeId) +
(SELECT COUNT(*)
FROM PIOutputDataYield t, inserted i
WHERE i.ioFlag != 0
AND t.pgeId = i.pgeId
AND t.logicalId = i.logicalId
AND t.dataTypeId = i.dataTypeId) != @numrows
BEGIN
    RAISERROR 81111 "No match for (pgeId, dataTypeId, logicalId) in table PIDataTypeReq
or PIOutputDataYield"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPIMetadataChecks

Trigger Code

```

CREATE TRIGGER TrigUpdPIMetadataChecks
ON PIIMetadataChecks
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

IF UPDATE ( pgeId ) OR UPDATE ( dataTypeId )
BEGIN

/* Verify that foreign key (pgeId, dataTypeId) has a matching
 * primary key in table PIIMetadataReq or PIOutputDataYield */

```

```

IF (SELECT COUNT(*)
     FROM PlDataTypeReq t, inserted i
     WHERE i.ioFlag = 0
       AND t.pgeId = i.pgeId
       AND t.logicalId = i.logicalId
       AND t.dataTypeId = i.dataTypeId) +
(SELECT COUNT(*)
     FROM PlOutputDataYield t, inserted i
     WHERE i.ioFlag != 0
       AND t.pgeId = i.pgeId
       AND t.logicalId = i.logicalId
       AND t.dataTypeId = i.dataTypeId) != @numrows
BEGIN
    RAISERROR 81112 "No match for (pgeId, dataTypeId, logicalId) in table
PlDataTypeReq or PlOutputDataYield"
    ROLLBACK TRANSACTION
    RETURN
END

END
RETURN

END
go

```

Trigger: TrigInsPlOutputDataYield

Trigger Code

```

CREATE TRIGGER TrigInsPlOutputDataYield
ON PlOutputDataYield
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

```

```

/* Verify that pgeId has a matching
 primary key in table PlPgeMaster */

IF (SELECT COUNT(*)
     FROM PlPgeMaster t, inserted i
      WHERE t.pgeId = i.pgeId)
    != @numrows
BEGIN
    RAISERROR 81111 "No match for pgeId in table PlPgeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that dataTypeId has a matching
 primary key in table PlDataTypeMaster */

IF (SELECT COUNT(*)
     FROM PlDataTypeMaster t, inserted i
      WHERE t.dataTypeId = i.dataTypeId)
    != @numrows
BEGIN
    RAISERROR 81111 "No match for dataTypeId in table PlDataTypeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPlOutputDataYield

Trigger Code

```

CREATE TRIGGER TrigUpdPlOutputDataYield
ON PlOutputDataYield
FOR UPDATE
AS

DECLARE @numrows int /* number of rows updated */

BEGIN

```

```

SELECT @numrows = @@rowcount

/* if no rows were deleted, then return */

IF @numrows = 0
    RETURN

/* see if the foreign keys were updated */

IF UPDATE ( pgeId )
BEGIN

    /* Verify that pgeId has a matching
     * primary key in table PlPgeMaster */

    IF (SELECT COUNT(*)
        FROM PlPgeMaster t, inserted i
        WHERE t.pgeId = i.pgeId)
        != @numrows
    BEGIN
        RAISERROR 81112 "No match for pgeId in table PlPgeMaster"
        ROLLBACK TRANSACTION
        RETURN
    END
END

IF UPDATE ( dataTypeId )
BEGIN

    /* Verify that dataTypeId has a matching
     * primary key in table PlDataTypeMaster */

    IF (SELECT COUNT(*)
        FROM PlDataTypeMaster t, inserted i
        WHERE t.dataTypeId = i.dataTypeId)
        != @numrows
    BEGIN
        RAISERROR 81112 "No match for dataTypeId in table PlDataTypeMaster"
        ROLLBACK TRANSACTION
        RETURN
    END
END

```

```

/* see if the primary key was updated */

IF UPDATE ( pgeId ) OR UPDATE ( dataTypeId ) OR UPDATE ( logicalId )
BEGIN

    /* update of pk on multiple rows not allowed */

    IF @numrows > 1
    BEGIN
        RAISERROR 81113 "Update to primary key on multiple rows is not supported"
        ROLLBACK TRANSACTION
        RETURN
    END

    /* cascade update to child tables */

    UPDATE PlMetadataChecks
    SET t.pgeId = i.pgeId,
        t.logicalId = i.logicalId,
        t.dataTypeId = i.dataTypeId
    FROM PlMetadataChecks t,
        inserted i, deleted d
    WHERE t.pgeId = d.pgeId
    AND t.dataTypeId = d.dataTypeId
    AND t.logicalId = d.logicalId
    AND t.ioFlag = 1

    IF @@transtate = 2 /* previous stmt aborted */
    BEGIN
        ROLLBACK TRANSACTION
        RETURN
    END

    UPDATE PlOutputFileType
    SET t.pgeId = i.pgeId,
        t.logicalId = i.logicalId,
        t.dataTypeId = i.dataTypeId
    FROM PlOutputFileType t,
        inserted i, deleted d
    WHERE t.pgeId = d.pgeId
    AND t.dataTypeId = d.dataTypeId
    AND t.logicalId = d.logicalId

```

```

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN

END
go

```

Trigger: TrigDelPlOutputDataYield

Trigger Code

```

CREATE TRIGGER TrigDelPlOutputDataYield
ON PlOutputDataYield
FOR DELETE
AS
BEGIN

/* if no rows were deleted, then return */

IF @@rowcount = 0
    RETURN

/* delete matching rows in PlMetadataChecks */

DELETE PlMetadataChecks
FROM PlMetadataChecks t, deleted d
WHERE t.pgeId = d.pgeId
AND t.dataTypeId = d.dataTypeId
AND t.logicalId = d.logicalId
AND t.ioFlag = 1

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlOutputFileType */

```

```

DELETE PIAssociatedScienceData
FROM PIAssociatedScienceData t, deleted d
WHERE t.pgeId = d.pgeId
AND t.dataTypeId = d.dataTypeId
AND t.logicalId = d.logicalId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

RETURN

END
go

```

Trigger: TrigInsPlPgeDetailParameters

Trigger Code

```

CREATE TRIGGER TrigInsPlPgeDetailParameters
ON PlPgeDetailParameters
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key has a matching
primary key in table PlPgeMaster */

IF (SELECT COUNT(*)
    FROM PlPgeMaster t, inserted i
    WHERE t.pgeId = i.pgeId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for pgeId in table PlPgeMaster"

```

```
ROLLBACK TRANSACTION
RETURN
END

RETURN

END
go
```

Trigger: TrigUpdPlPgeDetailParameters

Trigger Code

```
CREATE TRIGGER TrigUpdPlPgeDetailParameters
ON PlPgeDetailParameters
FOR UPDATE
AS
```

```
DECLARE @numrows int /* number of rows updated */

BEGIN

SELECT @numrows = @@rowcount

/* if no rows were deleted, then return */

IF @numrows = 0
    RETURN

/* see if the primary key was updated */

IF UPDATE ( pgeParameterLogicalId )
BEGIN

/* update of pk on multiple rows not allowed */

IF @numrows > 1
BEGIN
    RAISERROR 81113 "Update to primary key on multiple rows is not supported"
    ROLLBACK TRANSACTION
    RETURN
END
```

```

/* cascade update to child tables */

UPDATE PlPrParameter
SET t.pgeParameterLogicalId = i.pgeParameterLogicalId
FROM PlPrParameter t, inserted i, deleted d
WHERE t.pgeParameterLogicalId = d.pgeParameterLogicalId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

/* see if the foreign key was updated */

IF UPDATE ( pgeId )
BEGIN

    /* Verify that foreign key has a matching
     * primary key in table PlPgeMaster      */

    IF (SELECT COUNT(*)
        FROM PlPgeMaster t, inserted i
        WHERE t.pgeId = i.pgeId)
    != @numrows
    BEGIN
        RAISERROR 81112 "No match for pgeId in table PlPgeMaster"
        ROLLBACK TRANSACTION
        RETURN
    END

    END

    RETURN

END

go

```

Trigger: TrigDelPlPgeDetailParameters

Trigger Code

```
CREATE TRIGGER TrigDelPlPgeDetailParameters
ON PlPgeDetailParameters
FOR DELETE
AS
BEGIN

/* if no rows were deleted, then return */

IF @@rowcount = 0
    RETURN

/* delete matching rows in PlPrParameter */

DELETE PlPrParameter
FROM PlPrParameter t, deleted d
WHERE t.pgeParameterLogicalId = d.pgeParameterLogicalId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

RETURN

END
go
```

Trigger: TrigInsPlPgeDetailTile

Trigger Code

```
CREATE TRIGGER TrigInsPlPgeDetailTile
ON PlPgeDetailTile
FOR INSERT
AS
BEGIN
```

```

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key pgeId has a matching
   primary key in table PlPgeMaster      */

IF (SELECT COUNT(*)
    FROM PlPgeMaster t, inserted i
    WHERE t.pgeId = i.pgeId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for pgeId in table PlPgeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that foreign key schemaName has a matching
   primary key in table PlTileSchemaShort      */

IF (SELECT COUNT(*)
    FROM PlTileSchemaShort t, inserted i
    WHERE t.schemaName = i.schemaName)
< @numrows
BEGIN
    RAISERROR 81111 "No match for schemaName in table PlTileSchemaShort"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPlPgeDetailTile

Trigger Code

```
CREATE TRIGGER TrigUpdPlPgeDetailTile
ON PlPgeDetailTile
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

IF UPDATE ( pgId )
BEGIN

/* Verify that foreign key has a matching
 * primary key in table PlPgeMaster      */

IF (SELECT COUNT(*)
    FROM PlPgeMaster t, inserted i
    WHERE t.pgId = i.pgId)
!= @numrows
BEGIN
    RAISERROR 81112 "No match for pgId in table PlPgeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

END

IF UPDATE ( schemaName )
BEGIN

/* Verify that foreign key has a matching
 * primary key in table PlTileSchemaShort      */

IF (SELECT COUNT(*)
    FROM PlTileSchemaShort t, inserted i
    WHERE t.schemaName = i.schemaName)
!= @numrows
```

```

BEGIN
    RAISERROR 81112 "No match for schemaName in table PlTileSchemaShort"
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN

END
go

```

Trigger: TrigInsPlPgeDetailTime

Trigger Code

```

CREATE TRIGGER TrigInsPlPgeDetailTime
ON PlPgeDetailTime
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that every foreign key has a matching
primary key in table PlPgeMaster          */

IF (SELECT COUNT(*)
    FROM PlPgeMaster t, inserted i
    WHERE t.pgeId = i.pgeId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for pgeId in table PlPgeMaster"
    ROLLBACK TRANSACTION
    RETURN

```

```
END  
RETURN  
END  
go
```

Trigger: TrigUpdPlPgeDetailTime

Trigger Code

```
CREATE TRIGGER TrigUpdPlPgeDetailTime  
ON PlPgeDetailTime  
FOR UPDATE  
AS  
BEGIN  
  
/* See how many rows were updated */  
  
DECLARE @numrows int  
SELECT @numrows = @@rowcount  
IF @numrows = 0  
    RETURN  
  
IF UPDATE ( pgeId )  
BEGIN  
  
/* Verify that foreign key has a matching  
 * primary key in table PlPgeMaster */  
  
IF (SELECT COUNT(*)  
    FROM PlPgeMaster t, inserted i  
    WHERE t.pgeId = i.pgeId)  
!= @numrows  
BEGIN  
    RAISERROR 81112 "No match for pgeId in table PlPgeMaster"  
    ROLLBACK TRANSACTION  
    RETURN  
END  
  
END  
  
RETURN
```

```
END  
go
```

Trigger: TrigUpdPlPgeMaster

Trigger Code

```
CREATE TRIGGER TrigUpdPlPgeMaster  
ON PlPgeMaster  
FOR UPDATE  
AS  
  
DECLARE @numrows int /* number of rows updated */  
  
BEGIN  
  
SELECT @numrows = @@rowcount  
  
/* if no rows were deleted, then return */  
  
IF @numrows = 0  
    RETURN  
  
/* see if the primary key was updated */  
  
IF UPDATE ( pgeId )  
BEGIN  
  
/* update of pk on multiple rows not allowed */  
  
IF @numrows > 1  
BEGIN  
    RAISERROR 81113 "Update to primary key on multiple rows is not supported"  
    ROLLBACK TRANSACTION  
    RETURN  
END  
  
/* cascade update to child tables */  
  
UPDATE PlOutputDataYield  
SET t.pgeId = i.pgeId  
FROM PlOutputDataYield t,
```

```

    inserted i, deleted d
WHERE t.pgeId = d.pgeId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PlDataReq
SET t.pgeId = i.pgeId
FROM PlDataReq t,
     inserted i, deleted d
WHERE t.pgeId=d.pgeId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PlPgeDetailTile
SET t.pgeId = i.pgeId
FROM PlPgeDetailTile t, inserted i, deleted d
WHERE t.pgeId = d.pgeId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PlPgeDetailParameters
SET t.pgeId = i.pgeId
FROM PlPgeDetailParameters t, inserted i, deleted d
WHERE t.pgeId = d.pgeId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PlPgeDetailTime

```

```
SET t.pgeId = i.pgeId
FROM PlPgeDetailTime t, inserted i, deleted d
WHERE t.pgeId = d.pgeId
```

```
IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END
```

```
UPDATE PlProductionRequest
SET t.pgeIdentifier = i.pgeId
FROM PlProductionRequest t,
     inserted i, deleted d
WHERE t.pgeIdentifier = d.pgeId
```

```
IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END
```

```
UPDATE PlDataProcessingRequest
SET t.pgeId = i.pgeId
FROM PlDataProcessingRequest t,
     inserted i, deleted d
WHERE t.pgeId = d.pgeId
```

```
IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END
```

```
UPDATE PlPgePriority
SET t.pgeId = i.pgeId
FROM PlPgePriority t,
     inserted i, deleted d
WHERE t.pgeId = d.pgeId
```

```
IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN

```

```

END

UPDATE PlPgePerformance
SET t.pgeId = i.pgeId
FROM PlPgePerformance t,
     inserted i, deleted d
WHERE t.pgeId = d.pgeId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN

END
go

```

Trigger: TrigDelPlPgeMaster

Trigger Code

```

CREATE TRIGGER TrigDelPlPgeMaster
ON PlPgeMaster
FOR DELETE
AS
BEGIN

/* if no rows were deleted, then return */

IF @@rowcount = 0
    RETURN

/* delete matching rows in PlOutputDataYield */

DELETE PlOutputDataYield
FROM PlOutputDataYield t, deleted d
WHERE t.pgeId = d.pgeId

IF @@transtate = 2 /* previous statement aborted */
BEGIN

```

```

ROLLBACK TRANSACTION /* undo all deletes */
RETURN
END

/* delete matching rows in PlDataTypeReq */

DELETE PlDataTypeReq
FROM PlDataTypeReq t, deleted d
WHERE t.pgeId = d.pgeId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlPgeDetailTime */

DELETE PlPgeDetailTime
FROM PlPgeDetailTime t, deleted d
WHERE t.pgeId = d.pgeId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlPgeDetailParameters */

DELETE PlPgeDetailParameters
FROM PlPgeDetailParameters t, deleted d
WHERE t.pgeId = d.pgeId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlProductionRequest */

DELETE PlProductionRequest
FROM PlProductionRequest t, deleted d

```

```

WHERE t.pgeIdentifier = d.pgeId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlDataProcessingRequest */

DELETE PlDataProcessingRequest
FROM PlDataProcessingRequest t, deleted d
WHERE t.pgeId = d.pgeId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlPgePriority */

DELETE PlPgePriority
FROM PlPgePriority t, deleted d
WHERE t.pgeId = d.pgeId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlPgePerformance */

DELETE PlPgePerformance
FROM PlPgePerformance t, deleted d
WHERE t.pgeId = d.pgeId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

```

```

/* delete matching rows in PlPgeDetailTile */

DELETE PlPgeDetailTile
FROM PlPgeDetailTile t, deleted d
WHERE t.pgeId = d.pgeId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

RETURN

END
go

```

Trigger: TrigInsPlPgePerformance

Trigger Code

```

CREATE TRIGGER TrigInsPlPgePerformance
ON PlPgePerformance
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that every foreign key has a matching
primary key in table PlPgeMaster          */

IF (SELECT COUNT(*)
    FROM PlPgeMaster t, inserted i
    WHERE t.pgeId = i.pgeId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for pgeId in PlPgeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

```

```
RETURN
```

```
END
```

```
go
```

```
--
```

Trigger: TrigUpdPlPgePerformance

Trigger Code

```
CREATE TRIGGER TrigUpdPlPgePerformance
ON PlPgePerformance
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

IF UPDATE ( pgeId )
BEGIN

/* Verify that every foreign key has a matching
primary key in table PlPgeMaster */


IF (SELECT COUNT(*)
    FROM PlPgeMaster t, inserted i
    WHERE t.pgeId = i.pgeId)
!= @numrows
BEGIN
    RAISERROR 81112 "No match for pgeId in PlPgeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN
```

```
END  
go
```

Trigger: TrigInsPlPgePriority

Trigger Code

```
CREATE TRIGGER TrigInsPlPgePriority  
ON PlPgePriority  
FOR INSERT  
AS  
BEGIN  
  
/* See how many rows were inserted */  
  
DECLARE @numrows int  
SELECT @numrows = @@rowcount  
IF @numrows = 0  
    RETURN  
  
/* Verify that foreign key strategyId has a matching  
primary key in table PlProductionStrategy */  
  
IF (SELECT COUNT(*)  
    FROM PlProductionStrategy t, inserted i  
    WHERE t.strategyId = i.strategyId) != @numrows  
BEGIN  
    RAISERROR 81111 "No match for strategyId in table PlProductionStrategy"  
    ROLLBACK TRANSACTION  
    RETURN  
END  
  
RETURN  
  
END  
go
```

Trigger: TrigUpdPlPgePriority

Trigger Code

```
CREATE TRIGGER TrigUpdPlPgePriority
ON PlPgePriority
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

IF UPDATE ( strategyId )
BEGIN

/* Verify that foreign key strategyId has a matching
 * primary key in table PlProductionStrategy      */

IF (SELECT COUNT(*)
    FROM PlProductionStrategy t, inserted i
    WHERE t.strategyId = i.strategyId) != @numrows
BEGIN
    RAISERROR 81112 "No match for strategyId in table PlProductionStrategy"
    ROLLBACK TRANSACTION
    RETURN
END

END
END

RETURN

END
go
```

Trigger: TrigInsPlPlans

Trigger Code

```
CREATE TRIGGER TrigInsPlPlans
ON PlPlans
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key strategyId has a matching
primary key in table PlProductionStrategy      */

IF (SELECT COUNT(*)
    FROM PlProductionStrategy t, inserted i
    WHERE t.strategyId = i.strategyId) != @numrows
BEGIN
    RAISERROR 81111 "No match for strategyId in table PlProductionStrategy"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go
```

Trigger: TrigUpdPlPlans

Trigger Code

```
CREATE TRIGGER TrigUpdPlPlans
ON PlPlans
FOR UPDATE
```

AS

```
DECLARE @numrows int /* number of rows updated */

BEGIN

    SELECT @numrows = @@rowcount

    /* if no rows were deleted, then return */

    IF @numrows = 0
        RETURN

    /* see if the primary key was updated */

    IF UPDATE ( planId )
        BEGIN

            /* update of pk on multiple rows not allowed */

            IF @numrows > 1
                BEGIN
                    RAISERROR 81113 "Update to primary key on multiple rows is not supported"
                    ROLLBACK TRANSACTION
                    RETURN
                END

            /* cascade update to child tables */

            UPDATE PlPrVsPlan
                SET t.planId = i.planId
                FROM PlPrVsPlan t, inserted i, deleted d
                WHERE t.planId = d.planId

            IF @@transtate = 2 /* previous stmt aborted */
                BEGIN
                    ROLLBACK TRANSACTION
                    RETURN
                END
        END
    END

    /* see if the foreign key was updated */
```

```

IF UPDATE ( strategyId )
BEGIN

    /* Verify that foreign key strategyId has a matching
       primary key in table PIProductionStrategy      */

    IF (SELECT COUNT(*)
        FROM PIProductionStrategy t, inserted i
        WHERE t.strategyId = i.strategyId) != @numrows
    BEGIN
        RAISERROR 81112 "No match for strategyId in table PIProductionStrategy"
        ROLLBACK TRANSACTION
        RETURN
    END

    END

    RETURN

END
go

```

Trigger: TrigDelPIPlans

Trigger Code

```

CREATE TRIGGER TrigDelPIPlans
ON PIPlans
FOR DELETE
AS
BEGIN

    /* if no rows were deleted, then return */

    IF @@rowcount = 0
        RETURN

    /* delete matching rows in PIPrVsPlan */

    DELETE PIPrVsPlan
    FROM PIPrVsPlan t, deleted d
    WHERE t.planId = d.planId

```

```

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

RETURN

END
go

```

Trigger: TrigInsPlPrMetadataValue

Trigger Code

```

CREATE TRIGGER TrigInsPlPrMetadataValue
ON PlPrMetadataValue
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key productionRequestId has a matching
   primary key in table PlProductionRequest           */

IF (SELECT COUNT(*)
    FROM PlProductionRequest t, inserted i
    WHERE t.productionRequestId = i.productionRequestId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for productionRequestId in table PlPrMetadataValue"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

```

```
END  
go
```

Trigger: TrigUpdPlPrMetadataValue

Trigger Code

```
CREATE TRIGGER TrigUpdPlPrMetadataValue  
ON PlPrMetadataValue  
FOR UPDATE  
AS  
BEGIN  
  
/* See how many rows were updated */  
  
DECLARE @numrows int  
SELECT @numrows = @@rowcount  
IF @numrows = 0  
    RETURN  
  
IF UPDATE ( productionRequestId )  
BEGIN  
  
/* Verify that foreign key productionRequestId has a matching  
primary key in table PlProductionRequest */  
  
IF (SELECT COUNT(*)  
    FROM PlProductionRequest t, inserted i  
    WHERE t.productionRequestId = i.productionRequestId)  
!= @numrows  
BEGIN  
    RAISERROR 81112 "No match for productionRequestId in table PlPrMetadataValue"  
    ROLLBACK TRANSACTION  
    RETURN  
END  
  
END  
  
RETURN  
  
END  
go
```

Trigger: TrigInsPlProductionRequest

Trigger Code

```
CREATE TRIGGER TrigInsPlProductionRequest
ON PlProductionRequest
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that pgeId has a matching
primary key in table PlPgeMaster */

IF (SELECT COUNT(*)
    FROM PlPgeMaster t, inserted i
    WHERE t.pgeId = i.pgeIdentifier)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for pgeIdentifier in table PlPgeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go
```

Trigger: TrigUpdPlProductionRequest

Trigger Code

```
CREATE TRIGGER TrigUpdPlProductionRequest
ON PlProductionRequest
```

```

FOR UPDATE
AS

DECLARE @numrows int /* number of rows updated */

BEGIN

SELECT @numrows = @@rowcount

/* if no rows were deleted, then return */

IF @numrows = 0
    RETURN

/* see if the primary key was updated */

IF UPDATE ( productionRequestId )
BEGIN

/* update of pk on multiple rows not allowed */

IF @numrows > 1
BEGIN
    RAISERROR 81113 "Update to primary key on multiple rows is not supported"
    ROLLBACK TRANSACTION
    RETURN
END

/* cascade update to child tables */

UPDATE PlPrVsPlan
SET t.productionRequestId = i.productionRequestId
FROM PlPrVsPlan t, inserted i, deleted d
WHERE t.productionRequestId=d.productionRequestId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PlAlternateInputValues
SET t.productionRequestId = i.productionRequestId
FROM PlAlternateInputValues t,

```

```

    inserted i, deleted d
WHERE t.productionRequestId=d.productionRequestId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PIDataProcessingRequest
SET t.productionRequestId = i.productionRequestId
FROM PIDataProcessingRequest t,
     inserted i, deleted d
WHERE t.productionRequestId=d.productionRequestId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PIPrMetadataValue
SET t.productionRequestId = i.productionRequestId
FROM PIPrMetadataValue t, inserted i, deleted d
WHERE t.productionRequestId=d.productionRequestId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PIPrParameter
SET t.productionRequestId = i.productionRequestId
FROM PIPrParameter t, inserted i, deleted d
WHERE t.productionRequestId=d.productionRequestId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

END

```

```

/* see if foreign keys were updated */

IF UPDATE ( pgeIdentifier )
BEGIN

    /* Verify that pgeIdentifier has a matching
     * primary key in table PlPgeMaster */

    IF (SELECT COUNT(*)
        FROM PlPgeMaster t, inserted i
        WHERE t.pgeId = i.pgeIdentifier)
        != @numrows
    BEGIN
        RAISERROR 81112 "No match for pgeIdentifier in table PlPgeMaster"
        ROLLBACK TRANSACTION
        RETURN
    END
END

RETURN

END
go

```

Trigger: TrigDelPlProductionRequest

Trigger Code

```

CREATE TRIGGER TrigDelPlProductionRequest
ON PlProductionRequest
FOR DELETE
AS
BEGIN

    /* if no rows were deleted, then return */

    IF @@rowcount = 0
        RETURN

    /* delete matching rows in table PlPrVsPlan */

```

```

DELETE PIPrVsPlan FROM PIPrVsPlan t, deleted d
WHERE t.productionRequestId = d.productionRequestId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PIAlternateInputValues */

DELETE PIAlternateInputValues
FROM PIAlternateInputValues t, deleted d
WHERE t.productionRequestId = d.productionRequestId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PIDataProcessingRequest */

DELETE PIDataProcessingRequest
FROM PIDataProcessingRequest t, deleted d
WHERE t.productionRequestId = d.productionRequestId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PIPrMetadataValue */

DELETE PIPrMetadataValue
FROM PIPrMetadataValue t, deleted d
WHERE t.productionRequestId = d.productionRequestId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

```

```

/* delete matching rows in PIPrParameter */

DELETE PIPrParameter
FROM PIPrParameter t, deleted d
WHERE t.productionRequestId = d.productionRequestId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPIProductionStrategy

Trigger Code

```

CREATE TRIGGER TrigUpdPIProductionStrategy
ON PIProductionStrategy
FOR UPDATE
AS
BEGIN

/* if no rows were deleted, then return */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* see if the primary key was updated; if so, cascade the update
   to dependent tables */

IF UPDATE ( strategyId )
BEGIN

/* update of pk on multiple rows not allowed */

```

```

IF @numrows > 1
BEGIN
    RAISERROR 81113 "Update to primary key on multiple rows is not supported"
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PlPgePriority
SET t.strategyId = i.strategyId
FROM PlPgePriority t,
     inserted i, deleted d
WHERE t.strategyId = d.strategyId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PlUserPriority
SET t.strategyId = i.strategyId
FROM PlUserPriority t,
     inserted i, deleted d
WHERE t.strategyId = d.strategyId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PlPRPriority
SET t.strategyId = i.strategyId
FROM PlPRPriority t,
     inserted i, deleted d
WHERE t.strategyId = d.strategyId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PlPgePriority

```

```

SET t.strategyId = i.strategyId
FROM PIPgePriority t,
     inserted i, deleted d
WHERE t.strategyId = d.strategyId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PIPlans
SET t.strategyId = i.strategyId
FROM PIPlans t,
     inserted i, deleted d
WHERE t.strategyId = d.strategyId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN

END
go

```

Trigger: TrigDelPIProductionStrategy

Trigger Code

```

CREATE TRIGGER TrigDelPIProductionStrategy
ON PIProductionStrategy
FOR DELETE
AS
BEGIN

```

/* if no rows were deleted, then return */

```
IF @@rowcount = 0
```

```

RETURN

/* delete matching rows in PlPgePriority */

DELETE PlPgePriority
FROM PlPgePriority t, deleted d
WHERE t.strategyId = d.strategyId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlUserPriority */

DELETE PlUserPriority
FROM PlUserPriority t, deleted d
WHERE t.strategyId = d.strategyId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlPRPriority */

DELETE PlPRPriority
FROM PlPRPriority t, deleted d
WHERE t.strategyId = d.strategyId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlPlans */

DELETE PlPlans
FROM PlPlans t, deleted d
WHERE t.strategyId = d.strategyId

```

```

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

RETURN

END
go

```

Trigger: TrigInsPlPrParameter

Trigger Code

```

CREATE TRIGGER TrigInsPlPrParameter
ON PlPrParameter
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key productionRequestId has a matching
   primary key in table PlProductionRequest           */

IF (SELECT COUNT(*)
    FROM PlProductionRequest t, inserted i
    WHERE t.productionRequestId = i.productionRequestId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for productionRequestId in table PlProductionRequest"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that foreign key (pgeParameterLogicalId, pgeId) has a matching
   primary key in table PlPgeDetailParameters           */

```

```

IF (SELECT COUNT(*)
    FROM PlPgeDetailParameters t, inserted i
    WHERE t.pgeParameterLogicalId = i.pgeParameterLogicalId
    AND t.pgeId = i.pgeId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for pgeParameterLogicalId/pgeId in table
PlPgeDetailParameters"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPlPrParameter

Trigger Code

```

CREATE TRIGGER TrigUpdPlPrParameter
ON PlPrParameter
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

IF UPDATE ( productionRequestId )
BEGIN

/* Verify that foreign key productionRequestId has a matching
* primary key in table PlProductionRequest */

IF (SELECT COUNT(*)
    FROM PlProductionRequest t, inserted i

```

```

        WHERE t.productionRequestId = i.productionRequestId)
        != @numrows
    BEGIN
        RAISERROR 81112 "No match for productionRequestId in table PlProductionRequest"
        ROLLBACK TRANSACTION
        RETURN
    END

    END

    IF UPDATE ( pgeParameterLogicalId ) OR UPDATE ( pgeId )
    BEGIN

        /* Verify that foreign key (pgeParameterLogicalId, pgeId) has a matching
         * primary key in table PlPgeDetailParameters */
        IF (SELECT COUNT(*)
            FROM PlPgeDetailParameters t, inserted i
            WHERE t.pgeParameterLogicalId = i.pgeParameterLogicalId
            AND t.pgeId = i.pgeId)
        != @numrows
    BEGIN
        RAISERROR 81112 "No match for pgeParameterLogicalId/pgeId in table
PlPgeDetailParameters"
        ROLLBACK TRANSACTION
        RETURN
    END

    END

    RETURN

END
go

```

Trigger: TrigInsPlPRPriority

Trigger Code

```

CREATE TRIGGER TrigInsPlPRPriority
ON PlPRPriority
FOR INSERT
AS

```

```

BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key strategyId has a matching
   primary key in table PIProductionStrategy      */

IF (SELECT COUNT(*)
    FROM PIProductionStrategy t, inserted i
    WHERE t.strategyId = i.strategyId) != @numrows
BEGIN
    RAISERROR 81111 "No match for strategyId in table PIProductionStrategy"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPIPRPriority

Trigger Code

```

CREATE TRIGGER TrigUpdPIPRPriority
ON PIPRPriority
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

IF UPDATE ( strategyId )

```

```

BEGIN

/* Verify that foreign key strategyId has a matching
   primary key in table PlProductionStrategy      */

IF (SELECT COUNT(*)
    FROM PlProductionStrategy t, inserted i
    WHERE t.strategyId = i.strategyId) != @numrows
BEGIN
    RAISERROR 81112 "No match for strategyId in table PlProductionStrategy"
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN

END
go

```

Trigger: TrigInsPlPrVsPlan

Trigger Code

```

CREATE TRIGGER TrigInsPlPrVsPlan
ON PlPrVsPlan
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key productionRequestId has a matching
   primary key in table PlProductionRequest      */

IF (SELECT COUNT(*)
    FROM PlProductionRequest t, inserted i
    WHERE t.productionRequestId = i.productionRequestId)
    != @numrows
BEGIN

```

```

RAISERROR 81111 "No match for productionRequestId in table PIProductionRequest"
ROLLBACK TRANSACTION
RETURN
END

/* Verify that foreign key planId has a matching
   primary key in table PIPlans           */

IF (SELECT COUNT(*)
    FROM PIPlans t, inserted i
    WHERE t.planId = i.planId)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for planId in table PIPlans"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPIPrVsPlan

Trigger Code

```

CREATE TRIGGER TrigUpdPIPrVsPlan
ON PIPrVsPlan
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

IF UPDATE ( productionRequestId )
BEGIN

```

```

/* Verify that foreign key productionRequestId has a matching
 * primary key in table PIProductionRequest */
```

```

IF (SELECT COUNT(*)
    FROM PIProductionRequest t, inserted i
    WHERE t.productionRequestId = i.productionRequestId)
!= @numrows
BEGIN
    RAISERROR 81112 "No match for productionRequestId in table PIProductionRequest"
    ROLLBACK TRANSACTION
    RETURN
END
```

```

END
```

```

IF UPDATE ( planId )
BEGIN
```

```

/* Verify that foreign key planId has a matching
 * primary key in table PIPlans */
```

```

IF (SELECT COUNT(*)
    FROM PIPlans t, inserted i
    WHERE t.planId = i.planId)
!= @numrows
BEGIN
    RAISERROR 81112 "No match for planId in table PIPlans"
    ROLLBACK TRANSACTION
    RETURN
END
```

```

END
```

```

RETURN
```

```

END
go
```

Trigger: TrigInsPlRescUseThresh

Trigger Code

```
CREATE TRIGGER TrigInsPlRescUseThresh
ON PlRescUseThresh
FOR INSERT, UPDATE
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that resourceType is between 0 and 3 */

IF (SELECT COUNT(*)
    FROM inserted i
    WHERE i.resourceType BETWEEN 0 AND 3) != @numrows
BEGIN
    RAISERROR 81111 "Attempt to insert invalid resourceType into PlRescUseThresh"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that userType is either DSS_USER or else DPS_USER */

IF (SELECT COUNT(*)
    FROM inserted i
    WHERE (i.userType = "DSS_USER") OR (i.userType = "DPS_USER")) != @numrows
BEGIN
    RAISERROR 81111 "Attempt to insert invalid userType into PlRescUseThresh"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that, if resourceType is 3 (cumulative), then threshold is between 0 and 1 */

IF (SELECT COUNT(*)
    FROM inserted i
    WHERE i.resourceType = 3
    AND ((i.threshold < 0.0) OR (i.threshold > 1.0))) > 0
```

```

BEGIN
    RAISERROR 81111 "Attempt to insert invalid threshold for Cumulative into
PIRescUseThresh"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigInsPIRescUseThresh

Trigger Code

```

CREATE TRIGGER TrigInsPIRescUseThresh
ON PIRescUseThresh
FOR INSERT, UPDATE
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that resourceType is between 0 and 3 */

IF (SELECT COUNT(*)
    FROM inserted i
    WHERE i.resourceType BETWEEN 0 AND 3) != @numrows
BEGIN
    RAISERROR 81111 "Attempt to insert invalid resourceType into PIRescUseThresh"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that userType is either DSS_USER or else DPS_USER */

IF (SELECT COUNT(*)

```

```

    FROM inserted i
    WHERE (i.userType = "DSS_USER") OR (i.userType = "DPS_USER") ) != @numrows
BEGIN
    RAISERROR 81111 "Attempt to insert invalid userType into PIRescUseThresh"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that, if resourceType is 3 (cumulative), then threshold is between 0 and 1 */

IF (SELECT COUNT(*)
    FROM inserted i
    WHERE i.resourceType = 3
    AND ((i.threshold < 0.0) OR (i.threshold > 1.0)) ) > 0
BEGIN
    RAISERROR 81111 "Attempt to insert invalid threshold for Cumulative into
PIRescUseThresh"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPIResource

Trigger Code

```

CREATE TRIGGER TrigUpdPIResource
ON PIResource
FOR UPDATE
AS
BEGIN

/* if no rows were deleted, then return */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

```

```

IF UPDATE ( resourceId )
BEGIN

    /* update of pk on multiple rows not allowed */

    IF @numrows > 1
    BEGIN
        RAISERROR 81113 "Update to primary key on multiple rows is not supported"
        ROLLBACK TRANSACTION
        RETURN
    END

    UPDATE PIRscString
    SET t.stringId = i.resourceId
    FROM PIRscString t,
         inserted i, deleted d
    WHERE t.stringId = d.resourceId

    IF @@transtate = 2 /* previous stmt aborted */
    BEGIN
        ROLLBACK TRANSACTION
        RETURN
    END

    UPDATE PIRscComputer
    SET t.computerId = i.resourceId
    FROM PIRscComputer t,
         inserted i, deleted d
    WHERE t.computerId = d.resourceId

    IF @@transtate = 2 /* previous stmt aborted */
    BEGIN
        ROLLBACK TRANSACTION
        RETURN
    END

    UPDATE PIRscDiskPartition
    SET t.diskPartitionId = i.resourceId
    FROM PIRscDiskPartition t,
         inserted i, deleted d
    WHERE t.diskPartitionId = d.resourceId

    IF @@transtate = 2 /* previous stmt aborted */
    BEGIN

```

```

ROLLBACK TRANSACTION
RETURN
END

END

RETURN

END
go

```

Trigger: TrigDelPlResource

Trigger Code

```

CREATE TRIGGER TrigDelPlResource
ON PlResource
FOR DELETE
AS
BEGIN

/* if no rows were deleted, then return */

IF @@rowcount = 0
    RETURN

/* delete matching rows in table PlRscString */

DELETE PlRscString FROM PlRscString s,
                     deleted d
WHERE s.stringId = d.resourceId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

/* delete matching rows in table PlRscComputer */

DELETE PlRscComputer FROM PlRscComputer c,
                      deleted d
WHERE c.computerId = d.resourceId

```

```

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

/* delete matching rows in PIRscDiskPartition */

DELETE PIRscDiskPartition
FROM PIRscDiskPartition p, deleted d
WHERE p.diskPartitionId = d.resourceId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigInsPlResourceRequirement

Trigger Code

```

CREATE TRIGGER TrigInsPlResourceRequirement
ON PlResourceRequirement
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that column string, if not "" or null, has a matching
   value in column stringName in table PIRscString */

IF (SELECT COUNT(*)
    FROM PIRscString t, inserted i

```

```

        WHERE t.stringName = i.string) +
        (SELECT COUNT(*) FROM inserted i WHERE i.string = "") +
        (SELECT COUNT(*) FROM inserted i WHERE i.string is null)
    != @numrows
BEGIN
    RAISERROR 81111 "No match for string in table PlRscString"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that column computer, if not "" or null, has a matching
   value in column computerName in table PlRscComputer */

IF (SELECT COUNT(*)
    FROM PlRscComputer t, inserted i
    WHERE t.computerName = i.computer) +
    (SELECT COUNT(*) FROM inserted i WHERE i.computer = "") +
    (SELECT COUNT(*) FROM inserted i WHERE i.computer is null)
!= @numrows
BEGIN
    RAISERROR 81111 "No match for computer in PlRscComputer"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPlResourceRequirement

Trigger Code

```

CREATE TRIGGER TrigUpdPlResourceRequirement
ON PlResourceRequirement
FOR UPDATE
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount

```

```

IF @numrows = 0
    RETURN

IF UPDATE ( string )
BEGIN

    /* Verify that column string has a matching
     * value in column stringName in table PIRscString */

    IF ((SELECT COUNT(*)
        FROM PIRscString t, inserted i
        WHERE t.stringName = i.string) +
    (SELECT COUNT(*)
        FROM inserted i
        WHERE i.string = "") +
    (SELECT COUNT(*)
        FROM inserted i
        WHERE i.string is null))
        != @numrows
    BEGIN
        RAISERROR 81112 "No match for stringName in table PIRscString"
        ROLLBACK TRANSACTION
        RETURN
    END
END

IF UPDATE ( computer )
BEGIN

    /* Verify that column computer has a matching
     * value in column computerName in table PIRscComputer */

    IF ((SELECT COUNT(*)
        FROM PIRscComputer t, inserted i
        WHERE t.computerName = i.computer) +
    (SELECT COUNT(*)
        FROM inserted i
        WHERE i.computer = "") +
    (SELECT COUNT(*)
        FROM inserted i
        WHERE i.computer is null))
        != @numrows
    BEGIN

```

```

RAISERROR 81112 "No match for computerName in PIrscComputer"
ROLLBACK TRANSACTION
RETURN
END

END

RETURN

END
go

```

Trigger: TrigDelPIResourceRequirement

Trigger Code

```

CREATE TRIGGER TrigDelPIResourceRequirement
ON PIResourceRequirement
FOR DELETE
AS
BEGIN

/* if no rows were deleted, then return */

IF @@rowcount = 0
    RETURN

/* delete matching rows in PlPgeMaster */

DELETE PlPgeMaster
FROM PlPgeMaster t, deleted d
WHERE t.sswId = d.sswId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlDataProcessingRequest */

DELETE PlDataProcessingRequest

```

```

FROM PlDataProcessingRequest t, deleted d
WHERE t.sswId = d.sswId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in PlGroundEventAllocation */
/*
DELETE PlGroundEventAllocation
FROM PlGroundEventAllocation t, deleted d
WHERE t.sswId = d.sswId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END
*/
/* delete matching rows in DpPrPgeExitDependency */

DELETE DpPrPgeExitDependency
FROM DpPrPgeExitDependency t, deleted d
WHERE t.pgeSswId = d.sswId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

/* delete matching rows in DpPrPgeExitMessage */

DELETE DpPrPgeExitMessage
FROM DpPrPgeExitMessage t, deleted d
WHERE t.pgeSswId = d.sswId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

```

```
RETURN
```

```
END
```

```
go
```

Trigger: TrigInsPlRoutineArrival

Trigger Code

```
CREATE TRIGGER TrigInsPlRoutineArrival  
ON PlRoutineArrival  
FOR INSERT  
AS  
BEGIN
```

```
/* See how many rows were inserted */
```

```
DECLARE @numrows int
```

```
SELECT @numrows = @@rowcount
```

```
IF @numrows = 0
```

```
    RETURN
```

```
/* Verify that foreign key dataTypeId has a matching  
primary key in table PlDataSourceMaster */
```

```
IF (SELECT COUNT(*)
```

```
    FROM PlDataSourceMaster t, inserted i
```

```
    WHERE t.dataTypeId = i.dataTypeId) != @numrows
```

```
BEGIN
```

```
    RAISERROR 81111 "No match for dataTypeId in table PlDataSourceMaster"
```

```
    ROLLBACK TRANSACTION
```

```
    RETURN
```

```
END
```

```
RETURN
```

```
END
```

```
go
```

Trigger: TrigUpdPIRoutineArrival

Trigger Code

```
CREATE TRIGGER TrigUpdPIRoutineArrival
ON PIRoutineArrival
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

IF UPDATE ( dataTypeId )
BEGIN

/* Verify that foreign key dataTypeId has a matching
 * primary key in table PIDataSourceMaster      */

IF (SELECT COUNT(*)
    FROM PIDataSourceMaster t, inserted i
    WHERE t.dataTypeId = i.dataTypeId) != @numrows
BEGIN
    RAISERROR 81112 "No match for dataTypeId in table PIDataSourceMaster"
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN

END
go
```

Trigger: TrigInsPlRscComputer

Trigger Code

```
CREATE TRIGGER TrigInsPlRscComputer
ON PlRscComputer
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that key computerId has a matching
primary key in table PlResource      */

IF (SELECT COUNT(*)
    FROM PlResource t, inserted i
    WHERE t.resourceId = i.computerId) != @numrows
BEGIN
    RAISERROR 81111 "No match for computerId in table PlResource"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that foreign key stringId has a matching
primary key in table PlRscString      */

--IF (SELECT COUNT(*)
--FROM PlRscString t, inserted i
--WHERE t.stringId = i.stringId) != @numrows
--BEGIN
--    --RAISERROR 81111 "No match for stringId in table PlRscString"
--    --ROLLBACK TRANSACTION
--    --RETURN
--END

RETURN

END
go
```

Trigger: TrigUpdPlRscComputer

Trigger Code

```
CREATE TRIGGER TrigUpdPlRscComputer
ON PlRscComputer
FOR UPDATE
AS

DECLARE @numrows int /* number of rows updated */

BEGIN

SELECT @numrows = @@rowcount

/* if no rows were deleted, then return */

IF @numrows = 0
    RETURN

/* see if the primary key was updated */

IF UPDATE ( computerId )
BEGIN

/* update of pk on multiple rows not allowed */

IF @numrows > 1
BEGIN
    RAISERROR 81113 "Update to primary key on multiple rows is not supported"
    ROLLBACK TRANSACTION
    RETURN
END

/* cascade update to child tables */

UPDATE PlRscDiskPartition
SET t.computerId = i.computerId
FROM PlRscDiskPartition t, inserted i, deleted d
WHERE t.computerId = d.computerId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
```

```

ROLLBACK TRANSACTION
RETURN
END

END

/* see if foreign keys were updated */

IF UPDATE ( computerId )
BEGIN

/* Verify that computerId has a matching
 * primary key in table PlResource      */

IF (SELECT COUNT(*)
    FROM PlResource t, inserted i
    WHERE t.resourceId = i.computerId)
!= @numrows
BEGIN
    RAISERROR 81112 "No match for computerId in table PlResource"
    ROLLBACK TRANSACTION
    RETURN
END

END

IF UPDATE ( stringId )
BEGIN

/* Verify that stringId has a matching
 * primary key in table PlRscString      */

IF ((SELECT COUNT(*)
    FROM PlRscString t, inserted i
    WHERE t.stringId = i.stringId) +
    (SELECT COUNT(*)
        FROM inserted i
        WHERE i.stringId is null) +
    (SELECT COUNT(*)
        FROM inserted i
        WHERE i.stringId = 0))
!= @numrows
BEGIN
    RAISERROR 81112 "No match for stringId in table PlRscString"

```

```

ROLLBACK TRANSACTION
RETURN
END

END

RETURN

END
go

```

Trigger: TrigDelPlRscComputer

Trigger Code

```

CREATE TRIGGER TrigDelPlRscComputer
ON PlRscComputer
FOR DELETE
AS
BEGIN

/* if no rows were deleted, then return */

IF @@rowcount = 0
    RETURN

/* delete matching rows in PlRscDiskPartition */

DELETE PlRscDiskPartition
FROM PlRscDiskPartition c, deleted d
WHERE c.computerId = d.computerId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

/* delete matching rows in PlResource */

DELETE PlResource
FROM PlResource t, deleted d
WHERE t.resourceId = d.computerId

```

```

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigInsPlRscDiskPartition

Trigger Code

```

CREATE TRIGGER TrigInsPlRscDiskPartition
ON PlRscDiskPartition
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that key diskPartitionId has a matching
   primary key in table PlResource          */

IF (SELECT COUNT(*)
    FROM PlResource t, inserted i
    WHERE t.resourceId = i.diskPartitionId) != @numrows
BEGIN
    RAISERROR 81111 "No match for diskPartitionId in table PlResource"
    ROLLBACK TRANSACTION
    RETURN
END

/* Verify that foreign key computerId has a matching
   primary key in table PlRscComputer      */

```

```

--IF (SELECT COUNT(*)
    --FROM PlRscComputer t, inserted i
    --WHERE t.computerId = i.computerId) != @numrows
--BEGIN
    --RAISERROR 81111 "No match for computerId in table PlRscComputer"
    --ROLLBACK TRANSACTION
    --RETURN
--END

RETURN

END
go

```

Trigger: TrigUpdPlRscDiskPartition

Trigger Code

```

CREATE TRIGGER TrigUpdPlRscDiskPartition
ON PlRscDiskPartition
FOR UPDATE
AS

DECLARE @numrows int /* number of rows updated */

BEGIN

SELECT @numrows = @@rowcount

/* if no rows were deleted, then return */

IF @numrows = 0
    RETURN

/* see if the primary key was updated */

IF UPDATE ( diskPartitionId )
BEGIN

/* update of pk on multiple rows not allowed */

IF @numrows > 1

```

```

BEGIN
    RAISERROR 81113 "Update to primary key on multiple rows is not supported"
    ROLLBACK TRANSACTION
    RETURN
END

/* cascade update to child tables */

UPDATE DpPrDiskAllocation
SET t.diskPartitionId = i.diskPartitionId
FROM DpPrDiskAllocation t, inserted i, deleted d
WHERE t.diskPartitionId = d.diskPartitionId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

END

/* see if foreign keys were updated */

IF UPDATE ( computerId )
BEGIN

/* Verify that computerId has a matching
 * primary key in table PIRscComputer */

IF ((SELECT COUNT(*)
      FROM PIRscComputer t, inserted i
      WHERE t.computerId = i.computerId) +
    (SELECT COUNT(*)
      FROM inserted i
      WHERE i.computerId is null) +
    (SELECT COUNT(*)
      FROM inserted i
      WHERE i.computerId = 0))
!= @numrows

BEGIN
    RAISERROR 81112 "No match for computerId in table PIRscComputer"
    ROLLBACK TRANSACTION
    RETURN
END

```

```

END

IF UPDATE ( diskPartitionId )
BEGIN

    /* Verify that diskPartitionId has a matching
     * primary key in table PIResource      */

    IF (SELECT COUNT(*)
        FROM PIResource t, inserted i
        WHERE t.resourceId = i.diskPartitionId)
        != @numrows
    BEGIN
        RAISERROR 81112 "No match for diskPartitionId in table PIResource"
        ROLLBACK TRANSACTION
        RETURN
    END
END

RETURN

```

END

END

go

Trigger: TrigDelPIRscDiskPartition

Trigger Code

```

CREATE TRIGGER TrigDelPIRscDiskPartition
ON PIRscDiskPartition
FOR DELETE
AS
BEGIN

    /* if no rows were deleted, then return */

    IF @@rowcount = 0
        RETURN

    /* delete matching rows in DpPrDiskAllocation */

```

```

DELETE DpPrDiskAllocation
FROM DpPrDiskAllocation c, deleted d
WHERE c.diskPartitionId = d.diskPartitionId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

/* delete matching rows in PIResource */

DELETE PIResource
FROM PIResource t, deleted d
WHERE t.resourceId = d.diskPartitionId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigInsPIRscString

Trigger Code

```

CREATE TRIGGER TrigInsPIRscString
ON PIrscString
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

```

```

/* Verify that key stringId has a matching
   primary key in table PIResource      */

IF (SELECT COUNT(*)
    FROM PIResource t, inserted i
    WHERE t.resourceId = i.stringId) != @numrows
BEGIN
    RAISERROR 81111 "No match for stringId in PIResource"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPIRscString

Trigger Code

```

CREATE TRIGGER TrigUpdPIRscString
ON PIrscString
FOR UPDATE
AS

DECLARE @numrows int /* number of rows updated */

BEGIN

SELECT @numrows = @@rowcount

/* if no rows were deleted, then return */

IF @numrows = 0
    RETURN

/* see if the primary key was updated */

IF UPDATE ( stringId )
BEGIN

```

```

/* update of pk on multiple rows not allowed */

IF @numrows > 1
BEGIN
    RAISERROR 81113 "Update to primary key on multiple rows is not supported"
    ROLLBACK TRANSACTION
    RETURN
END

/* cascade update to child tables */

UPDATE PIRscComputer
SET t.stringId = i.stringId
FROM PIRscComputer t, inserted i, deleted d
WHERE t.stringId = d.stringId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

END

/* see if foreign keys were updated */

IF UPDATE ( stringId )
BEGIN

    /* Verify that stringId has a matching
     * primary key in table PIResource      */

    IF (SELECT COUNT(*)
        FROM PIResource t, inserted i
        WHERE t.resourceId = i.stringId)
        != @numrows
    BEGIN
        RAISERROR 81112 "No match for stringId in table PIResource"
        ROLLBACK TRANSACTION
        RETURN
    END

    END

```

```
RETURN  
END  
go
```

Trigger: TrigDelPIRscString

Trigger Code

```
CREATE TRIGGER TrigDelPIRscString  
ON PIRscString  
FOR DELETE  
AS  
BEGIN  
  
/* if no rows were deleted, then return */  
  
IF @@rowcount = 0  
    RETURN  
  
/* zero matching stringId in table PIRscComputer */  
  
UPDATE PIRscComputer  
SET c.stringId = 0  
FROM PIRscComputer c, deleted d  
WHERE c.stringId = d.stringId  
  
IF @@transtate = 2 /* previous statement aborted */  
BEGIN  
    ROLLBACK TRANSACTION  
    RETURN  
END  
  
/* delete matching rows in PIResource */  
  
DELETE PIResource  
FROM PIResource t, deleted d  
WHERE t.resourceId = d.stringId  
  
IF @@transtate = 2 /* previous statement aborted */  
BEGIN  
    ROLLBACK TRANSACTION  
    RETURN
```

```
END  
RETURN  
END  
go
```

Trigger: TrigUpdPlTileCoordinates

Trigger Code

```
CREATE TRIGGER TrigUpdPlTileCoordinates  
ON PlTileCoordinates  
FOR UPDATE  
AS  
  
DECLARE @numrows int /* number of rows updated */  
  
BEGIN  
  
SELECT @numrows = @@rowcount  
  
/* if no rows were deleted, then return */  
  
IF @numrows = 0  
    RETURN  
  
/* see if the primary key was updated */  
  
IF UPDATE ( tileId )  
BEGIN  
  
/* update of pk on multiple rows not allowed */  
  
IF @numrows > 1  
BEGIN  
    RAISERROR 81113 "Update to primary key on multiple rows is not supported"  
    ROLLBACK TRANSACTION  
    RETURN  
END  
  
/* cascade update to child tables */  
  
UPDATE PlTileSchemaShort
```

```

SET t.tileId = i.tileId
FROM PItileSchemaShort t, inserted i, deleted d
WHERE t.tileId = d.tileId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PIDataGranuleShort
SET t.tileId = i.tileId
FROM PIDataGranuleShort t, inserted i, deleted d
WHERE t.tileId = d.tileId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

UPDATE PIDataProcessingRequest
SET t.tileId = i.tileId
FROM PIDataProcessingRequest t, inserted i, deleted d
WHERE t.tileId = d.tileId

IF @@transtate = 2 /* previous stmt aborted */
BEGIN
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN

END
go

```

Trigger: TrigDelPlTileCoordinates

Trigger Code

```
CREATE TRIGGER TrigDelPlTileCoordinates
ON PlTileCoordinates
FOR DELETE
AS
BEGIN

/* if no rows were deleted, then return */

IF @@rowcount = 0
    RETURN

/* delete matching rows in PlTileSchemaShort */

DELETE PlTileSchemaShort
FROM PlTileSchemaShort t, deleted d
WHERE t.tileId = d.tileId

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    ROLLBACK TRANSACTION /* undo all deletes */
    RETURN
END

RETURN

END
go
```

Trigger: TrigInsPlTileSchemaShort

Trigger Code

```
CREATE TRIGGER TrigInsPlTileSchemaShort
ON PlTileSchemaShort
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
```

```

SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key tileId has a matching
   primary key in table PItileCoordinates */
```

```

IF (SELECT COUNT(*)
    FROM PItileCoordinates t, inserted i
    WHERE t.tileId = i.tileId) != @numrows
BEGIN
    RAISERROR 81111 "No match for tileId in table PItileCoordinates"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPItileSchemaShort

Trigger Code

```

CREATE TRIGGER TrigUpdPItileSchemaShort
ON PItileSchemaShort
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

IF UPDATE ( tileId )
BEGIN

/* Verify that foreign key tileId has a matching
   * primary key in table PItileCoordinates */
```

```

IF (SELECT COUNT(*)
     FROM PlTileCoordinates t, inserted i
     WHERE t.tileId = i.tileId) != @numrows
BEGIN
    RAISERROR 81112 "No match for tileId in table PlTileCoordinates"
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN

END
go

```

Trigger: TrigDelPlTileSchemaShort

Trigger Code

```

CREATE TRIGGER TrigDelPlTileSchemaShort
ON PlTileSchemaShort
FOR DELETE
AS
BEGIN

/* if no rows were deleted, then return */

IF @@rowcount = 0
    RETURN

/* Special delete: If this was the last surviving instance of tileId
   in this table, then delete tileId from PlTileCoordinates as well */

DELETE PlTileCoordinates
FROM  PlTileCoordinates t
WHERE t.tileId IN
    (SELECT tileId FROM deleted
     WHERE tileId NOT IN
        (SELECT tileId FROM PlTileSchemaShort))

RETURN

END

```

go

Trigger: TrigInsPlUsedByCenter

Trigger Code

```
CREATE TRIGGER TrigInsPlUsedByCenter
ON PlUsedByCenter
FOR INSERT
AS
BEGIN

/* See how many rows were inserted */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key dataTypeId has a matching
primary key in table PlDataTypeMaster */


IF (SELECT COUNT(*)
    FROM PlDataTypeMaster t, inserted i
    WHERE t.dataTypeId = i.dataTypeId) != @numrows
BEGIN
    RAISERROR 81111 "No match for dataTypeId in table PlDataTypeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go
```

Trigger: TrigUpdPlUsedByCenter

Trigger Code

```
CREATE TRIGGER TrigUpdPlUsedByCenter
ON PlUsedByCenter
```

```

FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

IF UPDATE ( dataTypeId )
BEGIN

/* Verify that foreign key dataTypeId has a matching
 * primary key in table PlDataTypeMaster      */

IF (SELECT COUNT(*)
    FROM PlDataTypeMaster t, inserted i
    WHERE t.dataTypeId = i.dataTypeId) != @numrows
BEGIN
    RAISERROR 81112 "No match for dataTypeId in table PlDataTypeMaster"
    ROLLBACK TRANSACTION
    RETURN
END

END

END

RETURN

END
go

```

Trigger: TrigInsPlUserPriority

Trigger Code

```

CREATE TRIGGER TrigInsPlUserPriority
ON PIUserPriority
FOR INSERT
AS
BEGIN

```

```

/* See how many rows were inserted */

```

```

DECLARE @numrows int

```

```

SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

/* Verify that foreign key strategyId has a matching
   primary key in table PlProductionStrategy      */

IF (SELECT COUNT(*)
    FROM PlProductionStrategy t, inserted i
    WHERE t.strategyId = i.strategyId) != @numrows
BEGIN
    RAISERROR 81111 "No match for strategyId in table PlProductionStrategy"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN

END
go

```

Trigger: TrigUpdPlUserPriority

Trigger Code

```

CREATE TRIGGER TrigUpdPlUserPriority
ON PlUserPriority
FOR UPDATE
AS
BEGIN

/* See how many rows were updated */

DECLARE @numrows int
SELECT @numrows = @@rowcount
IF @numrows = 0
    RETURN

IF UPDATE ( strategyId )
BEGIN

/* Verify that foreign key strategyId has a matching
   primary key in table PlProductionStrategy      */

```

```

IF (SELECT COUNT(*)
    FROM PIProductionStrategy t, inserted i
    WHERE t.strategyId = i.strategyId) != @numrows
BEGIN
    RAISERROR 81112 "No match for strategyId in table PIProductionStrategy"
    ROLLBACK TRANSACTION
    RETURN
END

END

RETURN

END
go

```

4.1.10 Stored Procedures

Sybase also includes support for business policy via the use of stored procedures. Stored procedures are typically used to capture a set of activities or checks that will be performed on the database repeatedly to enforce business policy and maintain data integrity. Stored procedures are parsed and compiled SQL code that reside in the database and may be called by name by an application, trigger or another stored procedure. A listing of the stored procedures in the PDPS database is given here. A brief definition of each of these stored procedures follows.

Procedure List

Name	Description
insertGroundEvent	Constructs a working ground event table based on information in resource planning tables.
ProcGetComments	Extracts comments pertaining to a particular resource in the database.
ProcGetResources	Extracts all resource objects from the database
ProcCleanup	performs database cleanup on Planning tables.
logDump-	Dumps the transaction log when a threshold is crossed.
logWarning	Issues a warning when a threshold is crossed.

Procedure: insertGroundEvent

Code

```

create proc insertGroundEvent
as

```

```

create table #ground_event
( name varchar(60) not null,
  descs varchar(30) null,
  start_time datetime not null,
  end_time datetime null,
  duration int null,
  priority int null,
  constraint pk_groun_event primary key (name, start_time)
)

insert #ground_event (name, descs, start_time, end_time, priority)
select rsv.name, rsv.description, intv.startTime, intv.stopTime, rsv.priority
from RpResourceReservation rsv, RpReservationInterval intv
where rsv.state = "approved"
and rsv.reservationId = intv.reservationId

update #ground_event
set duration = datediff(minute, start_time, end_time)
from #ground_event

select * from #ground_event
go

```

Procedure: ProcGetComments

Code

```

create proc ProcGetComments @commentId int, @commentType varchar(20)
as

select comment from RpRscPlanComments
where commentId = @commentId and
      commentType = @commentType
order by orderNum
go

```

Procedure: ProcGetResources

Code

```

create proc ProcGetResources as

create table #tmpRP
( rsId int not null,
  rsName varchar(60) null,

```

```

rsType varchar(15) null,
actName varchar(20) null,
stringId int null,
operSys varchar(60) null,
totRam int null,
totCpu int null,
computerId int null,
realComputerId int null,
partSize float null,
blockSize int null,
autosysIdKey int null)

insert #tmpRP(rsId, rsName, rsType, actName)
select rm.resourceId, rm.resourceName, rm.resourceType, ra.activityName
from PlResource rm, RpActivityType ra
where rm.activityTypeId *= ra.activityTypeId

update #tmpRP
set operSys = operatingSys,
totRam = totalRam,
totCpu = totalCpu,
#tmpRP.stringId = rc.stringId,
#tmpRP.realComputerId = rc.realComputerId
from PlRscComputer rc, #tmpRP
where rc.computerId = #tmpRP.rsId

update #tmpRP
set #tmpRP.autosysIdKey = rs.autosysIdKey
from PlRscString rs, #tmpRP
where rs.stringId = #tmpRP.rsId

update #tmpRP
set #tmpRP.computerId = rd.computerId, partSize = partitionSize,
#tmpRP.blockSize = rd.blockSize
from PlRscDiskPartition rd, #tmpRP
where rd.diskPartitionId = #tmpRP.rsId

select * from #tmpRP
go

```

Procedure: ProcCleanup

Code

```
CREATE PROC ProcCleanup
    (@day int,
     @mon int)

AS
BEGIN
BEGIN TRAN
/* Planning side clean up*/
DECLARE @date datetime
/* Subtract input month and day from current date */
SELECT @date = DATEADD(day, @day, dateadd(month, @mon, getdate()))
/*
* Clean up PlDataProcessingRequest table based on
* time stamp and completion state of dpr
*/
SELECT dprId FROM PlDataProcessingRequest
WHERE timeStamp < @date and completionState = "SUCC_DEL"
DELETE
PlDataProcessingRequest
WHERE timeStamp < @date and completionState = "SUCC_DEL"
/*
* Clean up PlProductionRequest
* Remove those PRs that doesn't have DPR any more.
*/
SELECT productionRequestId
FROM PlProductionRequest
WHERE NOT EXISTS
(SELECT 1 from PlDataProcessingRequest where PlProductionRequest.productionRequestId =
PlDataProcessingRequest.productionRequestId)
```

```

DELETE
PIProductionRequest
WHERE NOT EXISTS
(SELECT 1 from PIDataProcessingRequest where PIProductionRequest.productionRequestId =
PIDataProcessingRequest.productionRequestId)

/*
 * Clean up PIDataGranuleShort table based on
 * dynamicFlag(non-static), time stamp, no
 * dpr is using the granule and DPS is not using the granule
 */

SELECT granuleId
FROM PIDataGranuleShort, PIDataTypeMaster
WHERE PIDataGranuleShort.dataTypeId = PIDataTypeMaster.dataTypeId
AND PIDataGranuleShort.timeStamp < @date
AND PIDataTypeMaster.dynamicFlag != 0
AND NOT EXISTS
(SELECT 1 FROM PIprData WHERE PIDataGranuleShort.granuleId = PIprData.granuleId)
AND NOT EXISTS
(SELECT 1 FROM DpPrGranuleLocation WHERE DpPrGranuleLocation.granuleId =
PIDataGranuleShort.granuleId)

SELECT granuleId
FROM PIDataGranuleShort, PIDataTypeMaster
WHERE PIDataGranuleShort.dataTypeId = PIDataTypeMaster.dataTypeId
AND PIDataGranuleShort.timeStamp < @date
AND PIDataTypeMaster.dynamicFlag != 0
AND NOT EXISTS
(SELECT 1 FROM PIprData WHERE PIDataGranuleShort.granuleId = PIprData.granuleId)
AND EXISTS
(SELECT 1 FROM DpPrGranuleLocation WHERE DpPrGranuleLocation.granuleId =
PIDataGranuleShort.granuleId)

DELETE

```

```

PIDataGranuleShort
FROM PIDataGranuleShort, PIDataTypeMaster
WHERE PIDataGranuleShort.dataTypeId = PIDataTypeMaster.dataTypeId
AND PIDataGranuleShort.timeStamp < @date
AND PIDataTypeMaster.dynamicFlag != 0
AND NOT EXISTS
(SELECT 1 FROM PIdprData WHERE PIDataGranuleShort.granuleId = PIdprData.granuleId)
AND NOT EXISTS
(SELECT 1 FROM DpPrGranuleLocation WHERE DpPrGranuleLocation.granuleId =
PIDataGranuleShort.granuleId)

/*
 * Clean up PIpgeMaster table based on "deleteFlag" and no dprs for this PGE
 */

SELECT pgeId
FROM PIpgeMaster
WHERE PIpgeMaster.deleteFlag != 0
AND NOT EXISTS
(SELECT 1 FROM PIDataProcessingRequest WHERE PIDataProcessingRequest.pgeId =
PIpgeMaster.pgeId)

DELETE
FROM PIpgeMaster
WHERE PIpgeMaster.deleteFlag != 0
AND NOT EXISTS
(SELECT 1 FROM PIDataProcessingRequest WHERE PIDataProcessingRequest.pgeId =
PIpgeMaster.pgeId)

/*
 * Clean up PIResourceRequirement table based on
 * no other PGE has the sswId.
 */

SELECT sswId
FROM PIResourceRequirement

```

```

WHERE NOT EXISTS
(SELECT 1 FROM PlPgeMaster WHERE PlPgeMaster.sswId = PlResourceRequirement.sswId)
DELETE
PlResourceRequirement
FROM PlResourceRequirement, PlPgeMaster
WHERE NOT EXISTS
(SELECT 1 FROM PlPgeMaster WHERE PlPgeMaster.sswId = PlResourceRequirement.sswId)
COMMIT
END
go

```

Procedure: logdump

Code

```

CREATE PROCEDURE logdump
    @dbname varchar(30),
    @segmentname varchar(30),
    @space_left int,
    @status int
AS
DECLARE @devname varchar(100),
    @before_size int,
    @after_size int,
    @before_time datetime,
    @after_time datetime,
    @error int,
    @date char(8),
    @time char(4)

```

```
/* get the time and log size just before the dump */

SELECT @before_time = getdate(), @before_size = reserved_pgs(id, doampg)
FROM sysindexes
WHERE sysindexes.name = "syslogs"
PRINT "LOG DUMP: database '%1!', threshold '%2!'", @dbname, @space_left
```

```
/* get current date and time in format suitable for a filename suffix */
```

```
SELECT @date = datename(yy,getdate())+
    right('0'+ltrim(convert(varchar(2),
    datepart(mm,getdate())))),2)+
    right('0'+ltrim(convert(varchar(2),
    datepart(dd,getdate())))),2)
```

```
SELECT @time = right('0'+ltrim(convert(char(2),
    datepart(hh,getdate())))),2)+
    right('0'+ltrim(convert(char(2),
    datepart(mi,getdate())))),2)
```

```
/* dump device is a file in a mode-dependent directory owned by sybase */
```

```
SELECT @devname = "/usr/ecs/${MODE}/COTS/sybase/sybase_dumps/"
+ @dbname + "_tran_dmp." + @date + @time
```

```
/* dump the transaction log to the specified device */
```

```
DUMP TRANSACTION @dbname TO @devname
SELECT @error = @@error
```

```

IF @error != 0
BEGIN
    PRINT "LOG DUMP ERROR: %1!", @error
    RETURN @error
END

/* get size of log and time after dump */
SELECT @after_time= getdate(), @after_size =
    reserved_pgs(id, doampg)
FROM sysindexes
WHERE sysindexes.name = "syslogs"
/* print message to error log */
PRINT "LOG DUMPED TO: device '%1!', @devname
PRINT "LOG DUMP PAGES: Before: '%1!', After '%2!', "
    @before_size, @after_size
PRINT "LOG DUMP TIME: %1!, %2!",
    @before_time, @after_time

```

Procedure: logwarning

Code

```

CREATE PROCEDURE logwarning
    @dbname varchar(30),
    @segmentname varchar(30),
    @space_left int,
    @status int
AS
DECLARE @date datetime
SELECT @date = getdate()
PRINT "LOG WARNING: database '%1!', threshold '%2!', date/time '%3!', "
    @dbname, @space_left, @date

```

4.2 Flat File Usage

A flat file is an operating system file that is written and subsequently read, generally independent of other files that exist, and usually static in nature. There are cases when the implementation of persistent data is better suited to a flat file than to a database (e.g., system configuration data, external interface data). PDPS Subsystem file usage is detailed in this section.

4.2.1 File Descriptions

A summary listing of the files in the PDPS Subsystem is given in Table 4-6 together with a brief description of the file usage. Many different record formats are used in ECS including ODL, HDF, HDF EOS, block, fixed length, variable length, etc.

Table 4-6. Flat File Descriptions

File Name	File Type	Record Format	File Description
PGE ODL template	text	ODL	template for science metadata ODL file

4.2.2 Field Specifications

Brief specifications of the fields present within the PDPS Subsystem flat files are contained in Table 4-7. The fields are ordered alphabetically by File Name.

Table 4-7. Flat File Field Specifications (1 of 4)

File Name/ Block Name	Field Name	Data Type	Field Description
PGE ODL	PGE_NAME	string (max 10)	PGE name
PGE ODL	PGE_VERSION	string (max 5)	PGE version
PGE ODL	PROFILE_ID	integer between 0 and 99	PGE profile identifier

Table 4-7. Flat File Field Specifications (2 of 4)

File Name/ Block Name	Field Name	Data Type	Field Description
PGE ODL	PROFILE_DESCRIPTION	string (max 255)	PGE profile description
PGE ODL	PLATFORM	string (max 20)	Spacecraft platform name
PGE ODL	INSTRUMENT	string (max 10)	Instrument name
PGE ODL	MINIMUM_OUTPUTS	integer (max 3 digits)	Minimum number of outputs
PGE ODL	SCHEDULE_TYPE	"Time", "Data", "Tile", "Orbit", "Snapshot"	Type of PGE scheduling
PGE ODL	PROCESSING_PERIOD	String of the form "P=V" where P must be one of {YEARS, MONTHS, THIRDS, WEEKS, DAYS, HOURS, MINS, SECS, ORBITS}	Nominal time interval between start of PGE runs. e.g. "DAYS=1"
PGE ODL	PROCESSING_BOUNDARY	String of the form P(+ or - n) where n is an optional number of integer seconds. P must be one of {START_OF_HOUR, START_OF_6HOUR, START_OF_DAY, START_OF_WEEK, START_OF_ONE_THIRD_MONTH, START_OF_MONTH, START_OF_YEAR, START_DATE, START_OF_ORBIT}	Nominal time boundary on which PGE processing begins. Not needed for PGEs where SCHEDULE_TYPE = "Snapshot" or "Data". For PGEs based on an orbit model, this value will always be START_OF_ORBIT .
PGE ODL	PGE_SSW_VERSION	String (max 5)	Software version
PGE ODL	QUERY_DELAY	Integer > 0	Amount of time in seconds that the query for input data should be delayed.
PGE ODL	TILE_SCHEME_NAME	String (max 20), spaces not permitted	Name of the tiling scheme used.
PGE ODL	PATHMAP_NAME	String (max 25), spaces not permitted	Name of pathmap used.
PGE ODL	EXIT_MESSAGE	Object comprised of CLASS, EXIT_CODE, and EXIT_MESSAGE	Exit message object: defines a possible PGE exit code and associates a message with it.

Table 4-7. Flat File Field Specifications (3 of 4)

File Name/ Block Name	Field Name	Data Type	Field Description
PGE ODL	CLASS	Integer > 0	An object counter used to distinguish objects.
PGE ODL	EXIT_CODE	Integer = 0 or between 200 and 239	Exit code for this PGE.
PGE ODL	EXIT_MESSAGE	String (max 240)	Message corresponding to an exit code.
PGE ODL	EXIT_DEPENDENCY	Object comprised of DEPENDENCY_ PGE_NAME, DEPENDENCY_ SSW_VERSION, EXIT_OPERATION, EXIT_CODE	Exit dependency object
PGE ODL	DEPENDENCY_ PGE_NAME	String (max 10)	Name of PGE upon which this PGE depends.
PGE ODL	DEPENDENCY_ SSW_VERSION	String (max 5)	Version of SSW on which this SSW depends.
PGE ODL	EXIT_OPERATION	One of { >, <, >=, <=, =, != }	Operator for exit code dependency condition.

Table 4-7. Flat File Field Specifications (4 of 4)

File Name/ Block Name	Field Name	Data Type	Field Description
PGE ODL	PCF_ENTRY	<p>Object comprised of CLASS, LOGICAL_ID, PCF_FILE_TYPE, DATA_TYPE_NAME, DATA_TYPE_VERSION, MIN_GRANULES_REQUIRED, MAX_GRANULE_REQUIRED, BEGIN_PERIOD_OFFSET, END_PERIOD_OFFSET, SCIENCE_GROUP, INPUT_TYPE, NUMBER_NEEDED, DISTINCT_VALUE, QUERY_TYPE, SPATIAL_TIME_DELTA, KEY_INPUT, FILETYPE (class, filetype_name), ALTERNATE_INPUT (class, category, order, runtime_parm_id, timer, waitfor, temporal), OPTIONAL_INPUT(class, category, order, runtime_parm_id, timer, temporal), METADATA_CHECKS (class, parm_name, operator, value, database_query), METADATA_QUERY(class, parm_name, operator, value, database_query). Optional additional fields are MIN_GRANULE_YIELD, MAX_GRANULE_YIELD, ASSOCIATED_MCF_ID, INSTANCE, DISTINCT_VALUE, MINIMUM_SIZE, MAXIMUM_SIZE. Support input and output types may also have DATA_TYPE_NAME, DATA_TYPE_VERSION, DATA_TYPE_REQUIREMENT, PGE_PARAMETER_NAME, PGE_PARAMETER_DEFAULT, PGE_PARAMETER_DYNAMIC_VALUE. An interim or intermediate input will also have INTERIM_LAST_PGE_TO_USE.</p>	Possible entries in a process control file.

5. Performance and Tuning Factors

5.1 Indexes

An index provides a means of locating a row in a table based on the value of specific columns, without having to scan each row in the table. If used appropriately, indexes can significantly increase data retrieval. Sybase allows the definition of two types of indexes, clustered and non-clustered. In a clustered index, the rows in a table are physically stored in the sort order determined by the index. Clustered indexes are particularly useful when the data is retrieved in sort order. Non-clustered indexes differ from their clustered counterpart in that data is not physically stored in sort order. Only one clustered index may be defined per table. All of the indexes defined against tables in the PDPS database are described herein.

Index List

Table Code	Index Code	Primary Key	Foreign Key	Unique	Clustered
DpPrAutosysMapList	PK_DPPRAUTOSYSMAPLIST	Yes	No	Yes	Yes
DpPrAutosysMapList	DpPrAutosysMapList_OC	No	No	No	No
DpPrCreationQueue	PK_DPPRCREATIONQUEUE	Yes	No	Yes	Yes
DpPrDiskAllocation	DpPrDiskAllocation_OC1	No	No	No	No
	DpPrDiskAllocation_OC2	No	Yes	No	No
	DpPrDiskAllocation_OC3	No	No	No	No
DpPrDprDependList	PK_DPPRDPRDEPENDLIST	Yes	No	Yes	Yes
DpPrRpcID	PK_DPPRRPCID	Yes	No	Yes	Yes
DpPrExecutable	PK_DPPREEXECUTABLE	Yes	No	Yes	Yes
DpPrFile	PK_DPPRFIE	Yes	No	Yes	Yes
DpPrFile	DpPrFile_OC	No	No	No	No
DpPrGranuleLocation	PK_DPPRGANULELOCATION	Yes	No	Yes	Yes
DpPrGranuleLocation	DpPrGranuleLocation_FK1	No	No	No	No
DpPrPcf	PK_DPPRPCF	Yes	No	Yes	Yes
DpPrPge	PK_DPPRPGE	Yes	No	Yes	Yes
DpPrPge	DpPrPge_FK	No	Yes	No	No
	DpPrPge_OC	No	No	No	No
DpPrPgeExitDependency	PK_DPPRPGEEXITDEPENDENCY	Yes	No	Yes	Yes
DpPrPgeExitMessage	PK_DPPRPGEEXITMESSAGE	Yes	No	Yes	Yes
DpPrResourceLock	PK_DPPRRESOURCELOCK	Yes	No	Yes	Yes
DpPrResourceLock	DpPrResourceLock_OC1	No	No	No	No
	DpPrResourceLock_OC2	No	No	No	No
DpPrRscCpuRamAllocation	PK_DPPRRSCCPURAMALLOCATION	Yes	No	Yes	Yes
DpPrRscCpuRamAllocation	DpPrRscCpuRamAllocation_OC	No	No	No	No
PIAlternate	PK_PLALTERNATE	Yes	No	Yes	Yes

Table Code	Index Code	Primary Key	Foreign Key	Unique	Clustered
PIAlternateInputValues	PK_PLALTERNATEINPUTVALUES	Yes	No	Yes	Yes
PIAlternateInputValues	PIAlternateInputValues_OC	No	No	No	No
PIDataGranuleShort	PK_PLDATAGRANULESHORT	Yes	No	Yes	Yes
PIDataGranuleShort	DATA_TYPE_GRANULE_FK1 DATA_TYPE_GRANULE_FK2 DATA_TYPE_GRANULE_START_TIME DATA_TYPE_GRANULE_STOP_TIME PIDataTypeGranule_UI	No No No No No	Yes Yes No No No	No No No No Yes	No No No No No
PIDataProcessingRequest	PK_PLDATAPROCESSINGREQUEST	Yes	No	Yes	Yes
PIDataProcessingRequest	DATA_PROCESSING_REQUEST_FK1 DATA_PROCESSING_REQUEST_FK2 DATA_PROCESSING_REQUEST_FK3 PIDataProcessingRequest_OC	No No No No	Yes Yes Yes No	No No No No	No No No No
PIDataSourceMaster	PK_DATASOURCEMASTER	Yes	No	Yes	Yes
PIDataTypeFiles	PK_DATATYPEFILES	Yes	No	Yes	Yes
PIDataTypeMaster	PK_DATATYPEMASTER	Yes	No	Yes	Yes
PIDataTypeMaster	PIDataTypeMaster_OC1 PIDataTypeMaster_OC2	No No	No No	No No	No No
PIDataTypeReq	PK_DATATYPEPEREQ	Yes	No	Yes	Yes
PIDataTypeReq	PIDataTypeReq_OC	No	No	No	No
PIDprCompletion	PK_PLDPRCOMPLETION	Yes	No	Yes	Yes
PIDprData	PK_PLDPRDATA	Yes	No	Yes	Yes
PIDprData	ALTERNATE_DATA_GRANULE_FK PIDprData_OC	No No	Yes No	No No	No No
PIEsdtParam	PK_PLESDTPARAM	Yes	No	Yes	Yes
PIEsdtParamValues	PK_PLESDTPARMVALUES	Yes	No	Yes	Yes
PIFileTypeReq	PK_PLFILETYPEPEREQ	Yes	No	Yes	Yes
PIFileTypeReq	PIFileTypeReq_FK	No	No	No	No
PIGroundEvent	PK_PLGROUNDDEVENT	Yes	No	Yes	Yes
PIGroundEventResourceExpl	PK_PLGROUNDDEVENTRESOURCEEXPLOS	Yes	No	Yes	Yes
PIMetadataChecks	PK_PLMETADATACHECKS	Yes	No	Yes	Yes
PINotification	PK_PLNOTIFICATION	Yes	No	Yes	Yes
PIOputDataYield	PK_PLOUTPUTDATAYIELD	Yes	No	Yes	Yes
PIOputDataYield	PIOputDataYield_OC	No	No	No	No
PIOputFileType	PK_PLOUTPUTFILETYPE	Yes	No	Yes	Yes
PIOputFileType	PIOputFileType_FK	No	Yes	No	No
PIPPathModel	PK_PLPATHMODEL	Yes	No	Yes	Yes
PIPgeDetailParameters	PK_PGEDETAILPARAMETERS	Yes	No	Yes	Yes
PIPgeDetailTile	PK_PGEDETAILTILE	Yes	No	Yes	Yes
PIPgeDetailTime	PK_PGEDETAILTIME	Yes	No	Yes	Yes
PIPgeMaster	PK_PGEMASTER	Yes	No	Yes	Yes

Table Code	Index Code	Primary Key	Foreign Key	Unique	Clustered
PIPgeMaster	PGE_TITLE PIPgeMaster_OC	No No	No No	No No	No No
PIPgeOrbitModel	PK_PLPGEORBITMODEL	Yes	No	Yes	Yes
PIPgePerformance	PK_PLPGEPERFORMANCE	Yes	No	Yes	Yes
PIPgePriority	PK_PLPGEPRORITY	Yes	No	Yes	Yes
PIPlans	PK_PLPLANS	Yes	No	Yes	Yes
PIPlans	PLAN_FK1	No	Yes	No	No
PIPrMetadataValue	PK_PLPRMETADATAVALUE	Yes	No	Yes	Yes
PIProductionRequest	PK_PLPPRODUCTIONREQUEST	Yes	No	Yes	Yes
PIProductionRequest	PRODUCTION_REQUEST_FK1	No	Yes	No	No
PIProductionStrategy	PK_PLPPRODUCTIONSTRATEGY	Yes	No	Yes	Yes
PIPrParameter	PK_PLPRPARAMETER	Yes	No	Yes	Yes
PIPrParameter	PR_PARAMETER_FK PIPrParameter_OC	No No	No No	No No	No No
PIPRPriority	PK_PLPRPRIORTY	Yes	No	Yes	Yes
PIPrVsPlan	PIPrVsPlan_OC1 PIPrVsPlan_OC2	No No	No No	No No	No No
PIResource	PK_PLRESOURCE	Yes	No	Yes	Yes
PIResource	PIResource_OC	No	No	No	No
PIResourceRequirement	PK_PLRESOURCEREQUIREMENT	Yes	No	Yes	Yes
PIRoutineArrival	PK_ROUTINEARRIVAL	Yes	No	Yes	Yes
PIRscComputer	PK_RSCCOMPUTER	Yes	No	Yes	Yes
PIRscComputer	COMPUTER_LIST_EXPLOSION_FK PIRscComputer_OK PIRscComputer_FK2	No No No	Yes No Yes	No Yes No	No No No
PIRscDiskPartition	PK_RSCDISKPARTITION	Yes	No	Yes	Yes
PIRscDiskPartition	PIRscDiskPartition_OC	No	Yes	No	No
PIRscRealComputer	PK_RSCREALCOMPUTER	Yes	No	Yes	Yes
PIRscString	PK_RSCSTRING	Yes	No	Yes	Yes
PIRscString	PIRscString_OK PIRscString_OC	No No	No No	Yes No	No No
PITileCoordinates	PK_PLTILECOORDINATES	Yes	No	Yes	Yes
PITileSchemaShort	PK_PLTILESCHEMASHORT	Yes	No	Yes	Yes
PITimer	PITimer_PK PITimer_OC	Yes No	No No	Yes No	No Yes
PIUsedByCenter	USED_BY_CENTER_FK1	No	Yes	No	No
PIUserPriority	PK_PLUSERPRIORITY	Yes	No	Yes	Yes
RpActivityType	PK_RPACTIVITYTYPE	Yes	No	Yes	Yes
RpActivityType	RpActivityType_OC	No	No	No	No
RpReservationInterval	PK_RPRESERVATIONINTERVAL	Yes	No	Yes	Yes

Table Code	Index Code	Primary Key	Foreign Key	Unique	Clustered
RpResourceReservation	rpresourcereservation_fk PK_RPRESOURCERESERVATION RpResourceReservation_Cluster	No No No	Yes No No	No Yes No	No No Yes
RpRscPlanComments	RP_RSCPLANCOMMENTS	Yes	No	Yes	Yes
RpRsvRscExplosion	RP_RSVRSCEXPLOSION	Yes	No	Yes	Yes

5.2 Segments

Sybase supports the definition of segments. A segment is a named pointer to a storage device or devices. Segments are used to manually place database objects onto particular storage devices.

5.3 Named Caches

A cache is a block of memory that is used by Sybase to house data pages that are currently being accessed. A named cache is a named block of memory that the SQL server can use to house frequently accessed tables. Assigning a table to cache causes it to be loaded into memory. This greatly increases performance by eliminating the time expense normally associated with disk i/o. The PDPS database currently does not make use of named caches.

6. Database Security

6.1 Initial Users

Upon initial installation the following users will have access to PDPS database. The level of access is limited to that associated with their assigned group and/or role. A complete definition of each of these groups and roles is given below.

EcDpAtCheckODL
EcDpAtCreateODLTemplate
EcDpAtGetMCF
EcDpAtInsertExeTarFile
EcDpAtInsertStaticFile
EcDpAtInsertTestFile
EcDpAtOpDbGui
EcDpAtRegisterPGE
EcDpAtSSAPGui
EcDpAtStageDAP
EcDpPrCompression
EcDpPrDM
EcDpPrDeletion
EcDpPrEM
EcDpPrGE
EcDpPrJobMgmt
EcDpPrJobMgmtClient
EcDpPrQaMonitorGUI
EcDpPrViewJobStates
EcPlPREditor
EcPlPREditor_IF
EcPlPREditor_SUB

EcPlProdStrat

EcPlRm

EcPlRpRe

EcPlRpRm

EcPlRpSi

EcPlRpTl

EcPlSubMgr

EcPlSubsEdit

EcPlTl

EcPlWb

pdpsUsers

6.2 Groups

Groups are a means of logically associating users with similar data access needs. Once a group has been defined, object and command permissions can be granted to that group. A user who is a member of a group inherits all of the permissions granted to that group. The PDPS is installed with one group, pdps. All of the users listed in section 6.1 are members of this group.

6.3 Roles

Roles were introduced in Sybase 10 to allow a structured means for granting users the permissions needed to perform standard database administration activities and also provide a means for easily identifying such users. There are six pre-defined roles that may be assigned to a user. A definition of each of these roles follows as well as a description of the types of activities that may be performed by each role.

System Administrator (sa_role) - This role is used to grant a specific user to permissions needed to perform standard system administrator duties including:

- installing SQL server and specific SQL server modules
- managing the allocation of physical storage
- tuning configuration parameters
- creating databases

Site Security Officer (sso_role) - This role is used to grant a specific user the permissions needed to maintain SQL server security including:

- adding server logins
- administrating passwords
- managing the audit system
- granting users all roles except sa_role

Operator (oper_role) - This role is used to grant a specific user the permissions needed to manage backup and recovery of the database including;

- dumping transactions and databases
- loading transactions and databases

Navigator (navigator_role) -This role is used to grant a specific user the permissions needed to manage the navigation server.

Replication (replication_role) - - This role is used to grant a specific user the permissions needed to manage the replication server.

Sybase Technical Support (sybase_ts_role) - This role is used to grant a specific user the permissions needed to perform database consistency checker (dbcc), a sybase supplied utility, commands that are considered outside of the realm of normal system administrator activities.

Replication (replication_role) - - This role is used to grant a specific user the permissions needed to manage the replication server.

dbo – The dbo is the owner of the PDPS database with the right to create and destroy objects and to grant access to those objects. The user name pdps_role is an alias for the dbo role.

6.4 Object Permissions

In the PDPS database, any member of group pdps may perform select, insert, update, or delete on any user table or execute any user-defined stored procedure. Only the pdps_role may create or drop tables or other database objects.

This page intentionally left blank.

7. Scripts

7.1 Installation Scripts

Scripts used to support installation of the PDPS database are described herein. These files may be found in the directory /ecs/formal/PDPS/PLS/sybase

Script File	Description
EcP IDbLogins	Adds login Ids needed by various PDPS clients and applications.
EcP IDbDrop	Drops all objects in the database.
EcP IDbBuild	Creates new database schema (selected within EcCoAssist)
EcP IDbPatch	Modifies a database schema (selected within EcCoAssist)
EcP ICreat ePDPSDatabase	Creates new database schema and establishes a list of valid users (invoked by EcP IDbBuild)
EcP IInitializePDPSDatabase	Populate tables needing initialization (invoked by EcP IDbBuild)

7.2 Backup/Recovery Scripts

Scripts used to facilitate backup or recovery of the PDPS database are described herein.

Script File	Description
EcP IDbDump	Dumps the PDPS planning database to a flat file that can be used to recover.
EcP IDbLoad	Recovers (loads) the PDPS planning database from a database dump flat file.

7.3 Miscellaneous Scripts

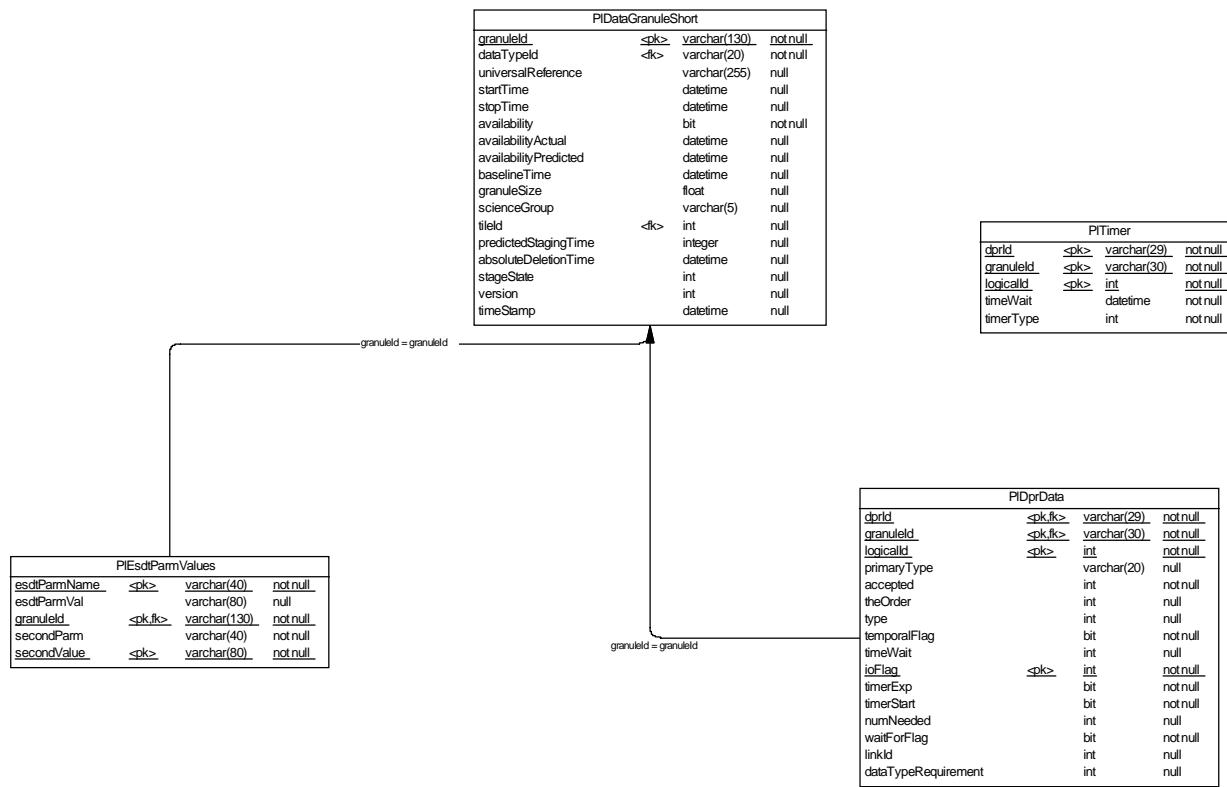
Miscellaneous scripts applicable to the PDPS database are described herein.

Script File	Description
EcP IDbStat	Updates the Sybase statistics for the PDPS planning database.

This page intentionally left blank.

Appendix - PDPS Database Schema Diagrams

Figures A-1 through A-13 represent the ERDs for the PDPS database.



Physical Data Model	
Project :	ECS_PDPS
Model :	Data_Granules
Author :	GD Version: 4P 7/20/98

Figure A-1. Data Granules

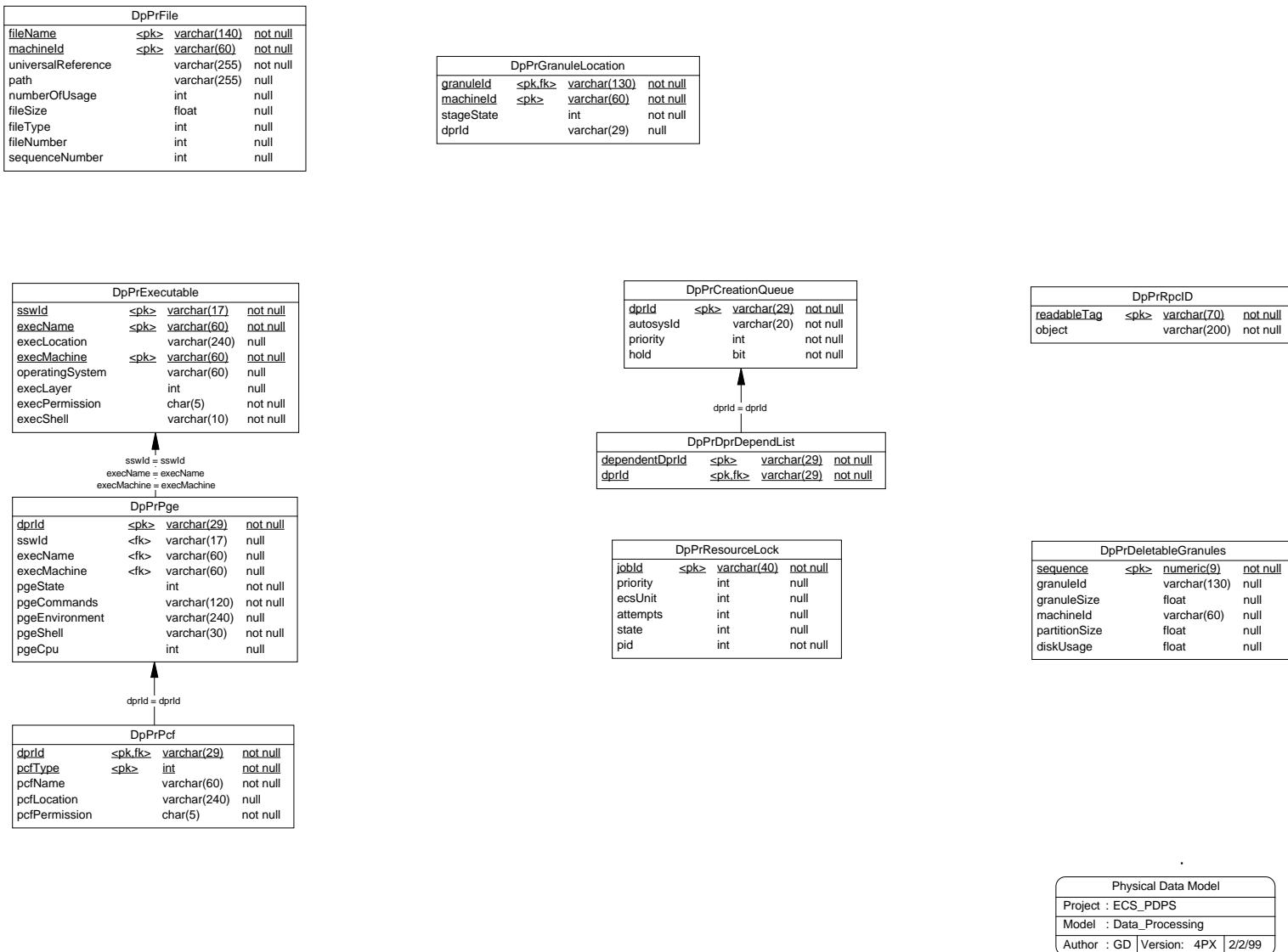
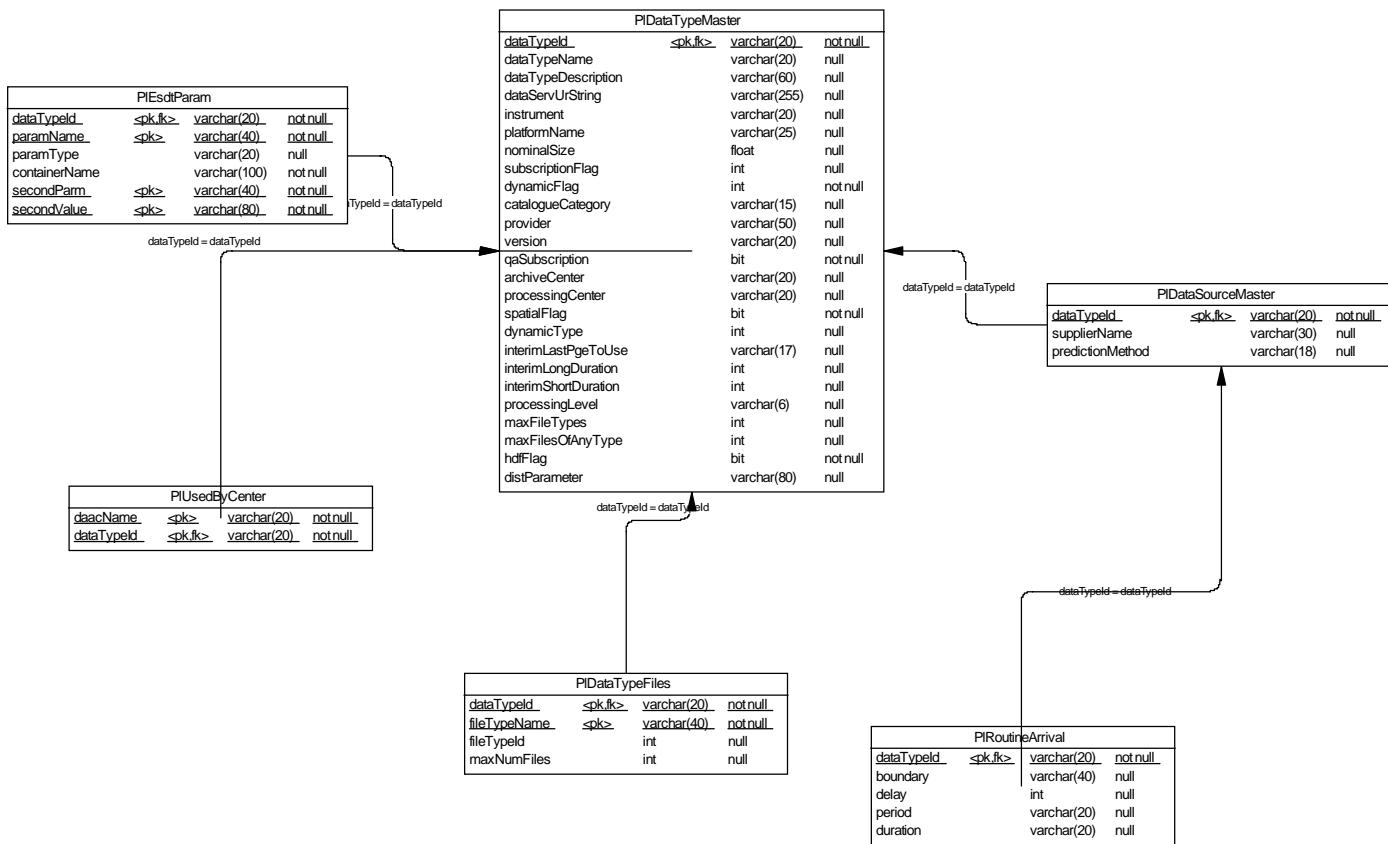


Figure A-2. Data Processing



Physical Data Model			
Project	: ECS_PDPS		
Model	: Data_Types		
Author	: GD	Version:	4P 10/29/98

Figure A-3. Data Types

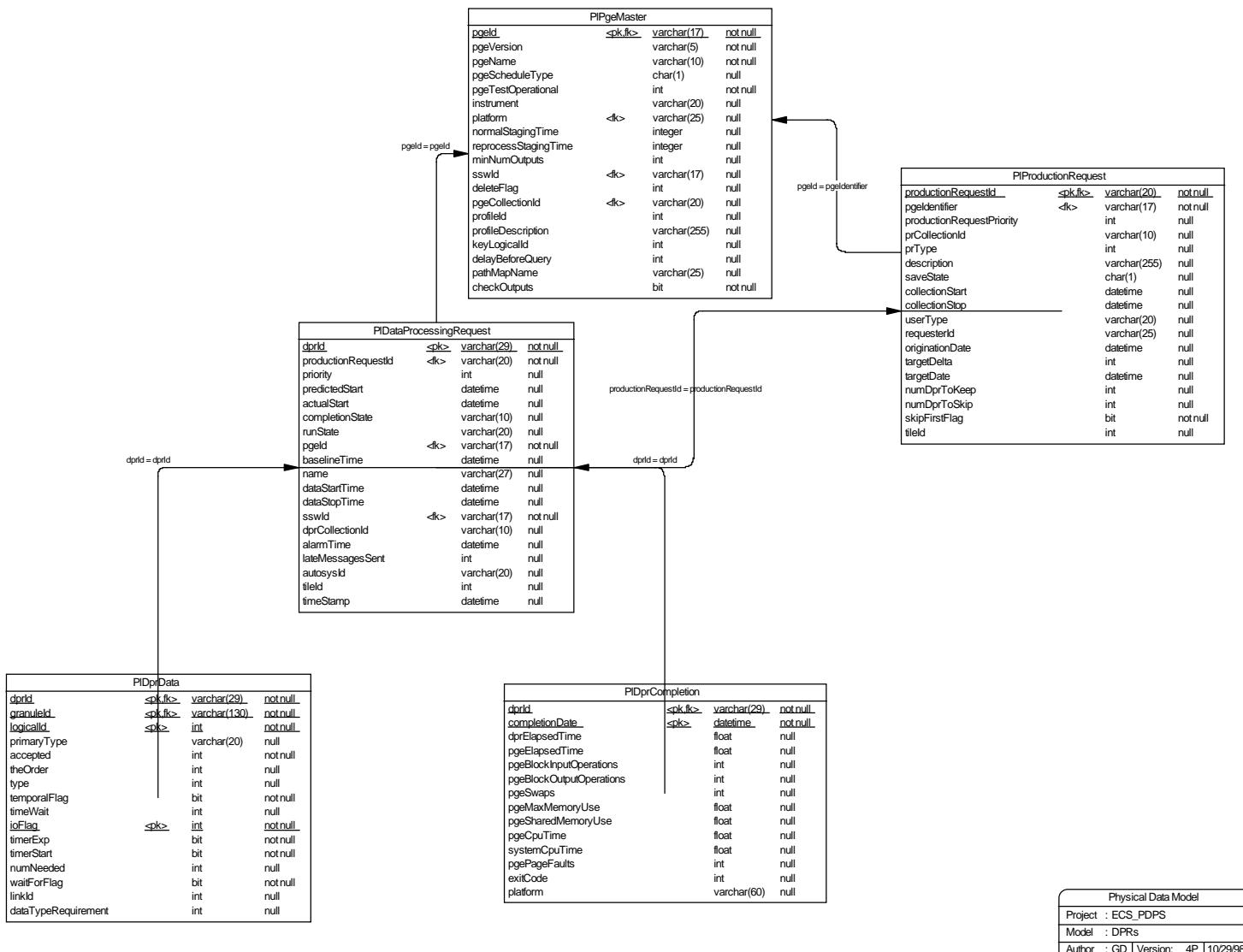
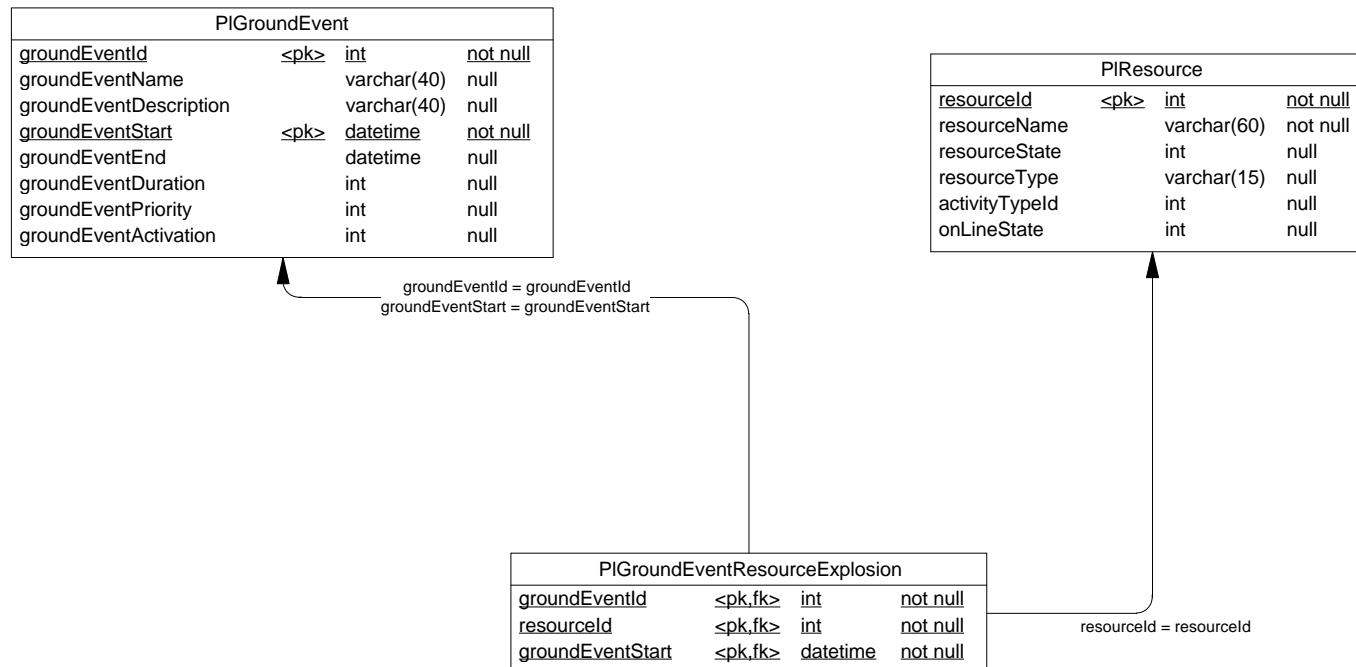
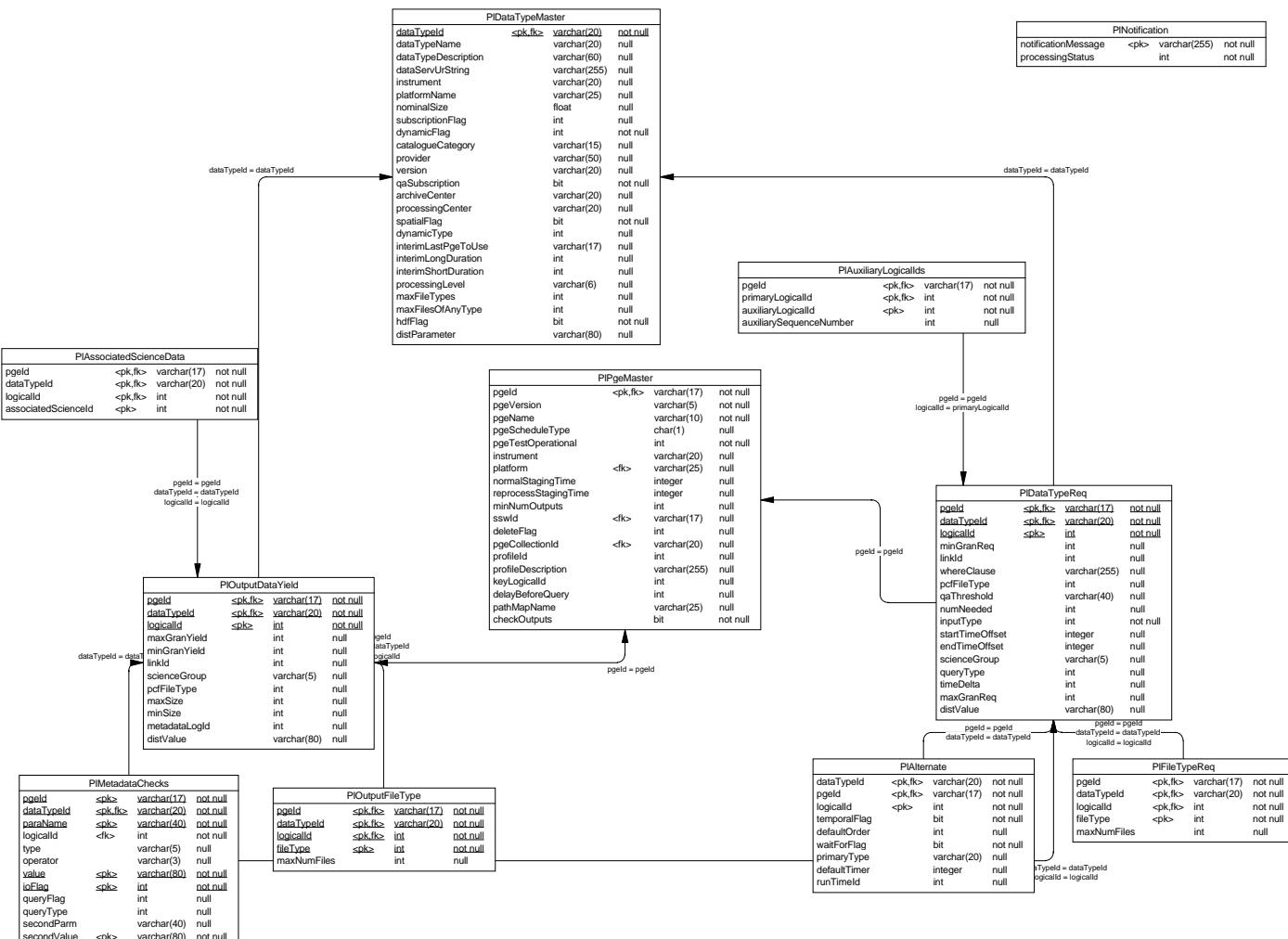


Figure A-4. Data Processing Requests



Physical Data Model	
Project :	ECS_PDPS
Model :	Ground_Events
Author :	GD Version: 4PX 2/2/99

Figure A-5. Ground Events



Physical Data Model	
Project :	ECS_POPS
Model :	PGE_Inputs_Outputs
Author :	GD Version: 4PX 2/2/99

Figure A-6. PGE Inputs & Outputs

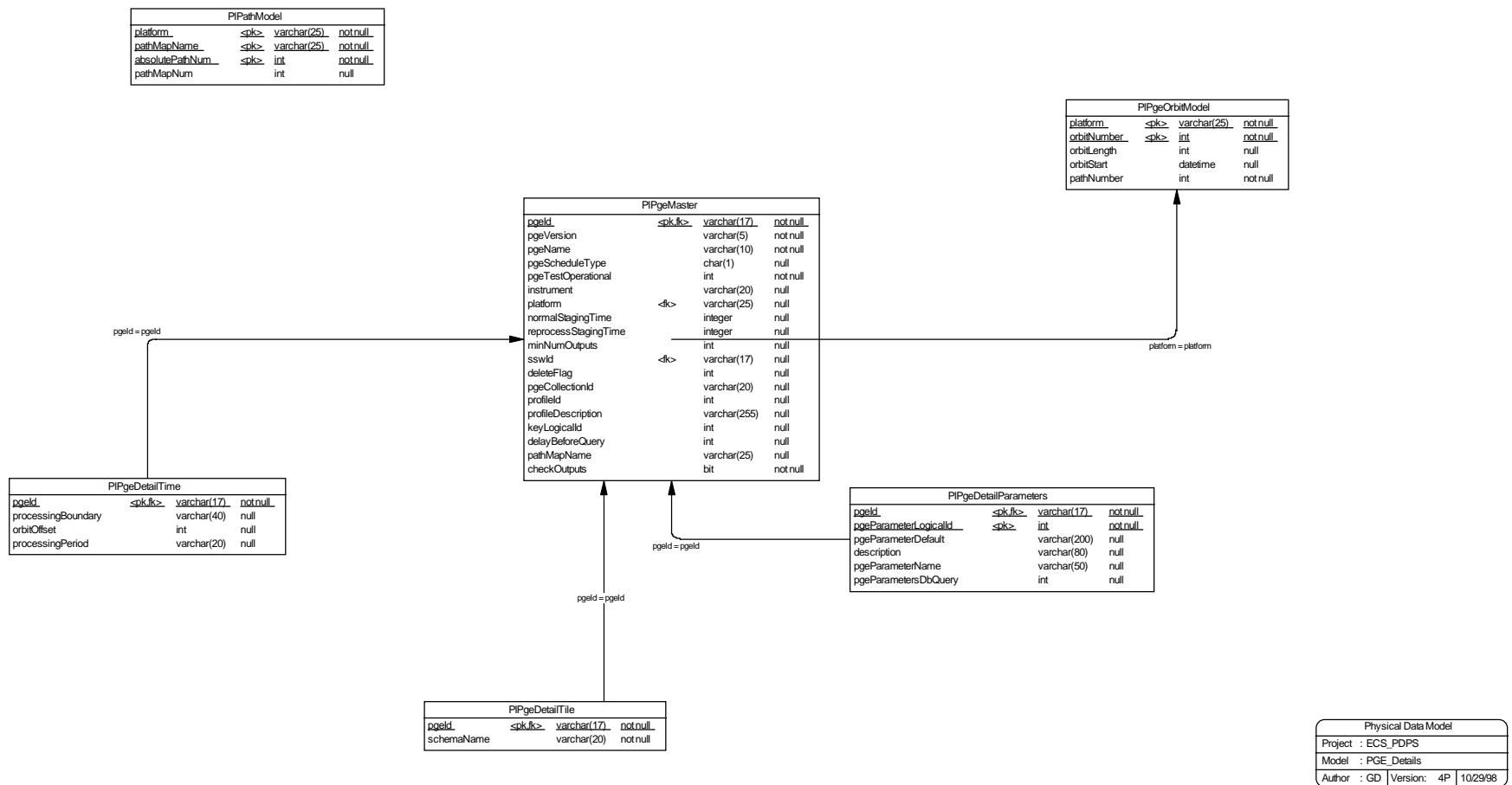
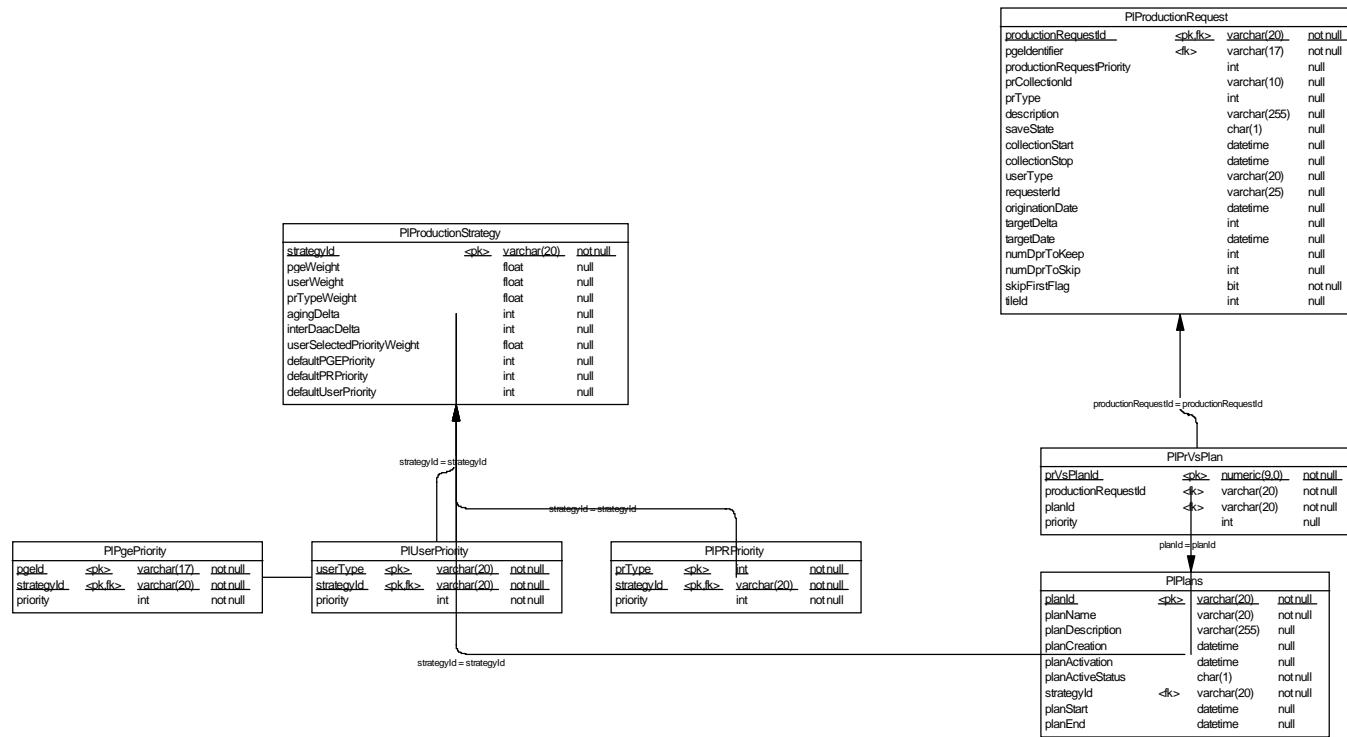


Figure A-7. PGE Details



Physical Data Model	
Project :	ECS_PDPS
Model :	Plan_Creation
Author :	GD Version: 4P 10/29/98

Figure A-8. Plan Creation

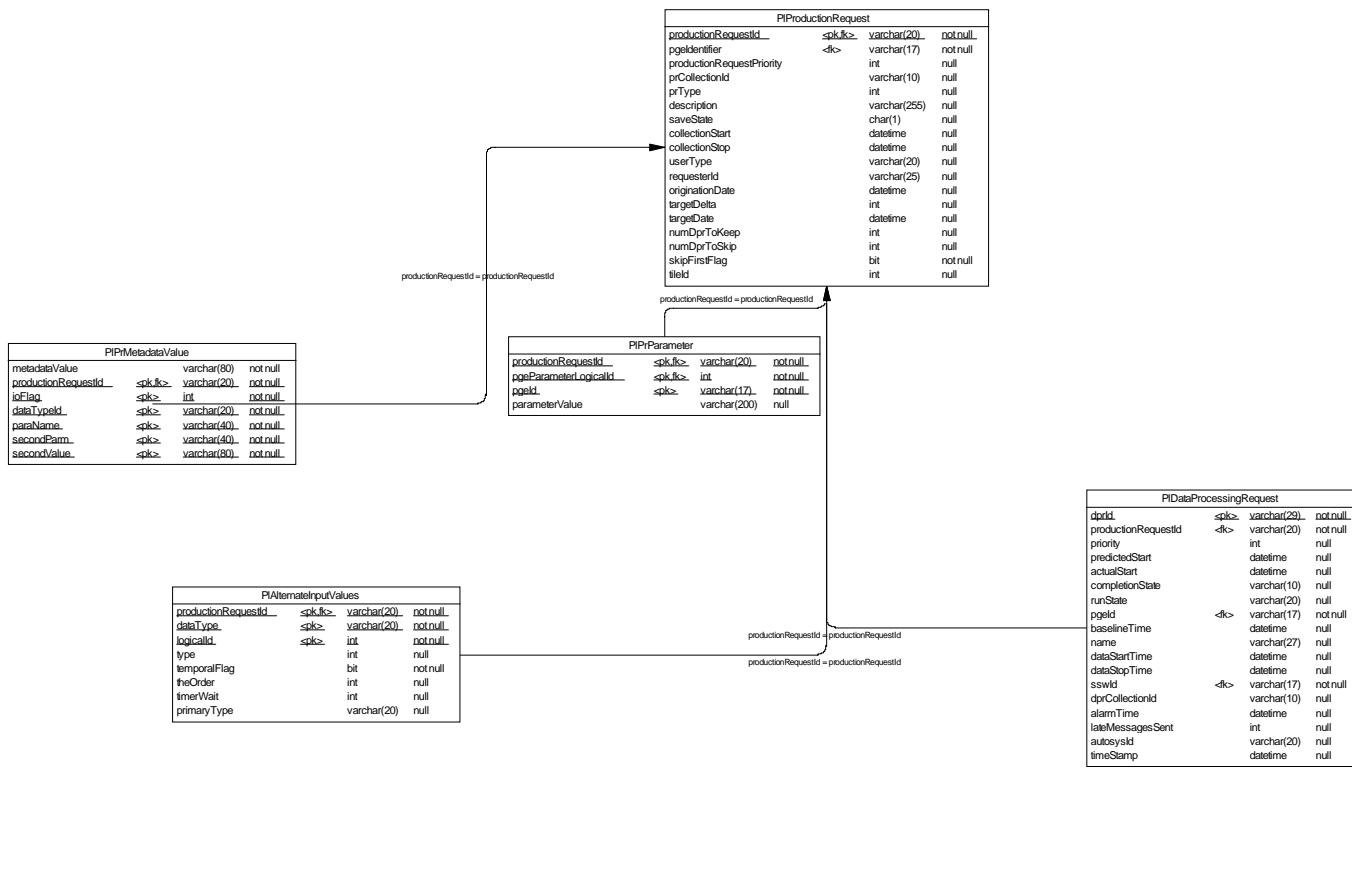


Figure A-9. Production Requests

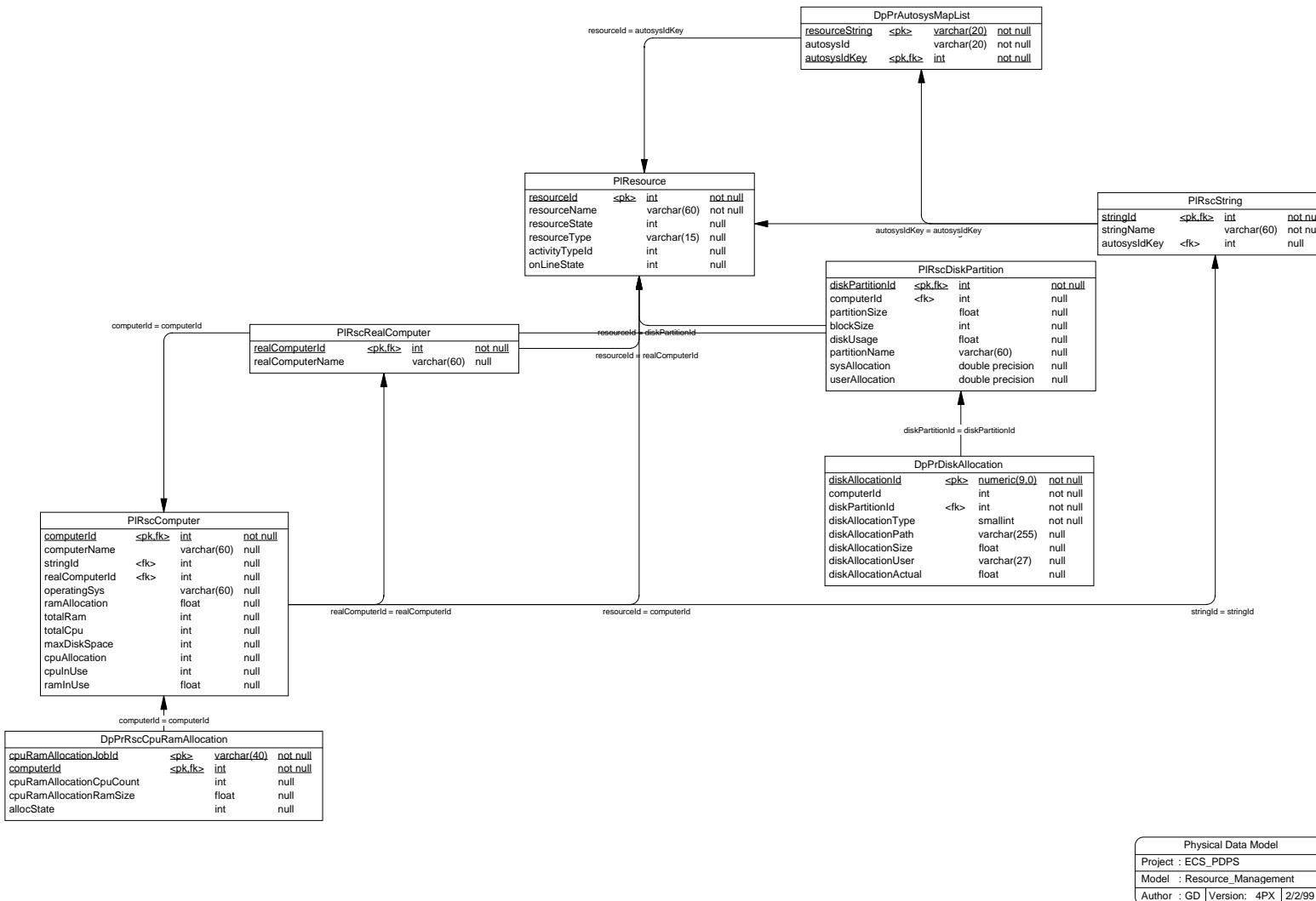
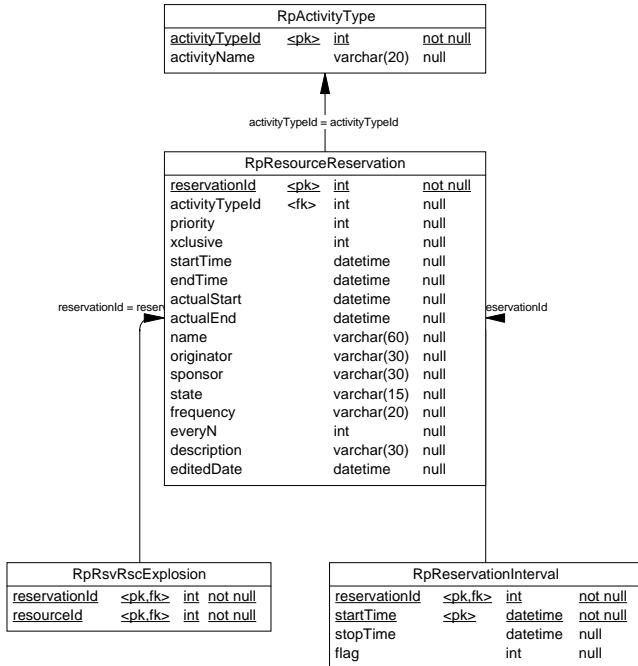


Figure A-10. Resource Management

RpRscPlanComments			
commentId	<pk>	int	not null
commentType	<pk>	varchar(20)	not null
orderNum	<pk>	int	not null
comment		varchar(255)	null



Physical Data Model			
Project :	ECS_PDPS		
Model :	Resource_Planning		
Author :	GD	Version:	4PX 2/2/99

Figure A-11. Resource Planning

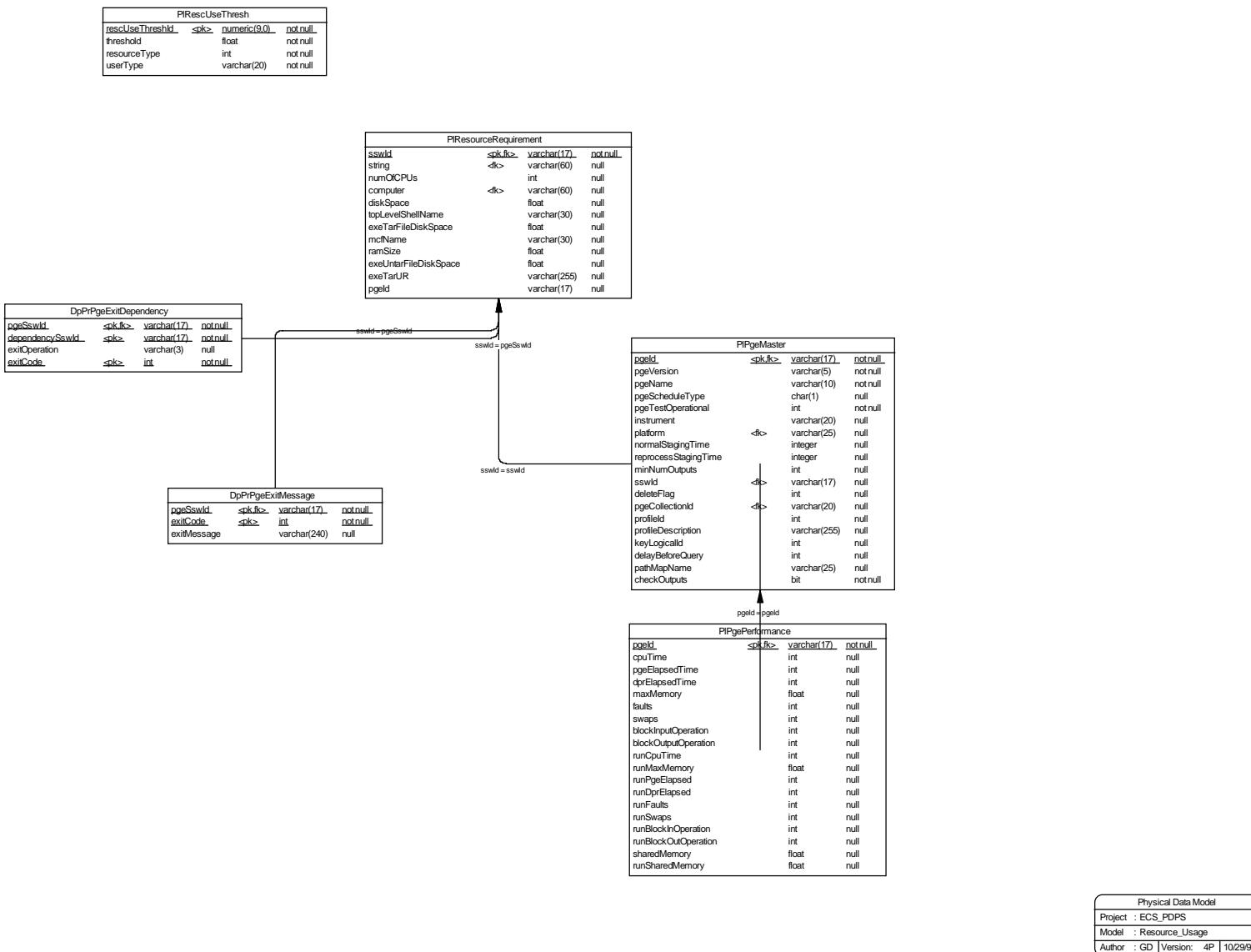
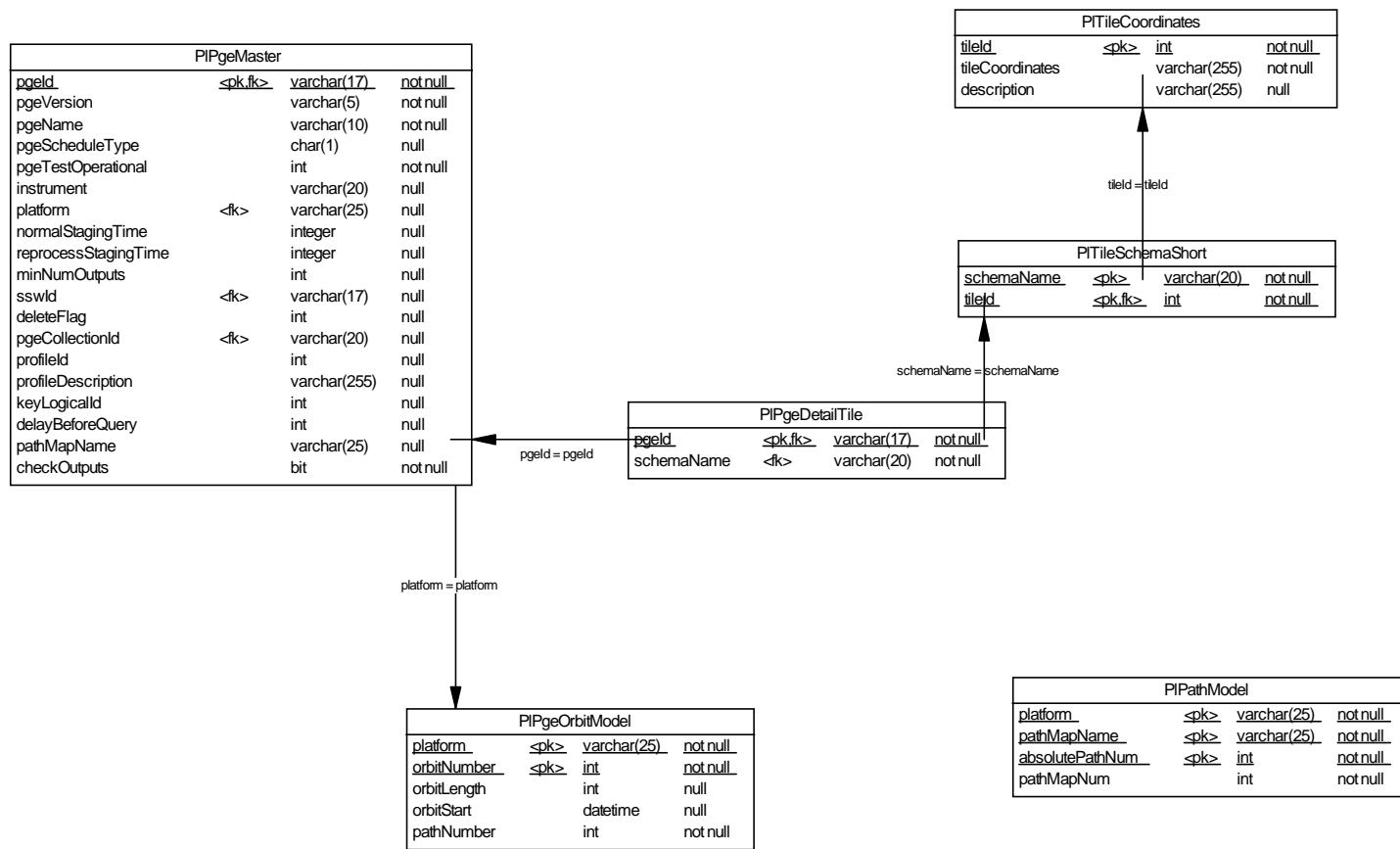


Figure A-12. Resource Usage



Physical Data Model		
Project :	ECS_PDPS	
Model :	Tiling	
Author :	GD	Version: 4PX 10/29/98

Figure A-13. Tiling

Abbreviations and Acronyms

ACL	Access Control List
ACMHW	Access and Control Management HWCI
ADC	Affiliated Data Center
ADSHW	Advertising Server HWCI
ADSRV	Advertising Service CSCI
AI&T	algorithm integration and test
AITHW	Algorithm Integration and Test HWCI
AITTL	Algorithm Integration and Test CSCI
AM-1	EOS AM Project spacecraft 1, morning spacecraft series -- ASTER, CERES, MISR, MODIS and MOPITT instruments
ANSI	American National Standards Institute
API	application program (or programming) interface
APID	application's process ID
AQAHW	Algorithm QA HWCI
ASCII	American Standard Code for Information Exchange
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer (formerly ITIR)
AVHRR	Advanced Very High-Resolution Radiometer
BER	bit error rate
BUFR	binary universal format for representation of data
CASE	Computer Aided Software Engineering
CCSDS	Consultative Committee for Space Data Systems
CD	contractual delivery 214-001
CD-ROM	compact disk -- read only memory
CDR	Critical Design Review
CDRL	contract data requirements list
CERES	Clouds and Earth's Radiant Energy System
CI	configuration item

COTS	commercial off-the-shelf (hardware or software)
CPU	central processing unit
CSCI	computer software configuration item
CSDT	Computer Science Data Type
CSMS	Communications and Systems Management Segment (ECS)
CSS	Communications Subsystem
DAAC	Distributed Active Archive Center
DAN	data availability notice
DAO	Data Assimilation Office
DAR	data acquisition request
DAS	data availability schedule
DBMS	Database Management System
DDICT	Data Dictionary CSCI
DDIST	Data Distribution Services CSCI
DDSRV	Document Data Server CSCI
DESKT	Desktop CSCI
DID	data item description
DIM	distributed information manager (SDPS)
DIMGR	Distributed Information Manager CSCI
DIPHW	Distribution and Ingest Peripheral Management HWCI
DMGHW	Data Management HWCI
DMS	Data Management Subsystem
DMWG	Data Management Working Group
DP	Data Provider
DPR	data processing request
DPREP	Science Data Preprocessing CSCI
DPS	Data Processing Subsystem
DRPHW	Data Repository HWCI
DSS	Data Server Subsystem
ECS	EOSDIS Core System

EDC	EROS Data Center
EDHS	ECS Data Handling System
EDOS	EOS Data and Operations System
EOS	Earth Observing System
EOS-AM	EOS Morning Crossing (Descending) Mission -- see AM-1
EOSDIS	Earth Observing System Data and Information System
EROS	Earth Resources Observation System
ESDIS	Earth Science Data and Information System (GSFC)
ESDT	Earth science data types
ESN	EOSDIS Science Network (ECS)
FDDI	fiber distributed data interface
FDF	flight dynamics facility
FDFEPHEM	FDF-generated definitive orbit data
FGDC	Federal Geographic Data Committee
FK	Foreign Key
FOO	Flight of Opportunity
FOS	Flight Operations Segment (ECS)
GB	gigabyte (10^9)
GNU	(recursive acronym: “GNU’s Not Unix”); a project supported by the Free Software Foundation dedicated to the delivery of free software
GPCP	Global Precipitation Climatology Project
GPCP	Global Precipitation Climatology Project
GPI	GOES Precipitation Index
GRIB	GRid In Binary
GSFC	Goddard Space Flight Center
GTWAY	Version 0 Interoperability Gateway CSCI
GUI	graphic user interface
GV	ground validation
HDF	hierarchical data format
HDF-EOS	an EOS proposed standard for a specialized HDF data format

HIPPI	high performance parallel interface
HMI	human machine interface
HTML	HyperText Markup Language
HTTP	Hypertext Transport Protocol
HWCI	hardware configuration item
I&T	integration and test
I/F	interface
I/O	input/output
ICD	interface control document
ICLHW	Ingest Client HWCI
ID	identification
IDE	Interactive Development Environments
IDG	Infrastructure Development Group
IDR	Incremental Design Review
IERS	International Earth Rotation Service
IMS	Information Management System (obsolete ECS element name)
INGST	Ingest Services CSCI
IOS	Interoperability Subsystem
IP	international partners
IR-1	Interim Release 1
IRD	interface requirements document
ISO	International Standards Organization
ISS	Internetworking Subsystem
IV&V	independent verification and validation
JPL	Jet Propulsion Laboratory
L0-L4	Level 0 (zero) through Level 4
LaRC	Langley Research Center (DAAC)
LIM	local information manager (SDPS)
LIMGR	Local Information Manager CSCI
LIS	Lightning Imaging Sensor

LSM	local system management (ECS)
MB	megabyte (10^6)
MDT	mean downtime
MDT	mean downtime
MFLOPS	mega (millions of) floating-point operations (10^6) per second
MISR	Multi-Angle Imaging SpectroRadiometer
MODIS	Moderate-Resolution Imaging Spectrometer
MOPITT	Measurements of Pollution in the Troposphere
MSFC	Marshall Space Flight Center
MSS	Management Support Subsystem
MTBF	mean time between failure
MTPE	Mission to Planet Earth
MTTR	mean time to restore
N/A	not applicable
NAS	National Academy of Science
NASA	National Aeronautics and Space Administration
NESDIS	National Environmental Satellite Data and Information Service
NMC	National Meteorological Center (NOAA)
NOAA	National Oceanic and Atmospheric Administration
NSIDC	National Snow and Ice Data Center (DAAC)
O/A	orbit/altitude
ODC	other data center
OSI	Open System Interconnect
PDPS	Planning and Data Processing Subsystem
PDR	Preliminary Design Review
PDS	production data set
PGE	Product Generation Executive
PGS	Product Generation System (obsolete ECS element name) (ASTER)
PK	Primary Key
PLANG	Production Planning CSCI

PLNHW	Planning HWCI
PLS	Planning Subsystem
POSIX	Portable Operating System Interface for Computer Environments
PR	Precipitation Radar (TRMM)
PRONG	Processing CSCI
QA	quality assurance
RMA	reliability, maintainability, availability
RTF	rich text format
SAA	satellite active archive
SAGE	Stratospheric Aerosol and Gas Experiment
SCF	Science Computing Facility
SDP	Science Data Processing
SDPF	Sensor Data Processing Facility (GSFC)
SDPS	Science Data Processing Segment (ECS)
SDPTK	SDP Toolkit CSCI
SDSRV	Science Data Server CSCI
SeaWIFS II	Sea-Viewing Wide Field-of-View Sensor II
SFDU	Standard Format Data Unit
SMC	System Management Center (ECS)
SPRHW	Science Processing HWCI
SRS	software requirements specification
SSM/I	Special Sensor for Microwave/Imaging (DMSP)
SST	sea surface temperature
STMGMT	Storage Management
STMGT	Storage Management Software CSCI
SUBSRV	Subscription Server
TMI	TRMM Microwave Image
TOMS	Total Ozone Mapping Spectrometer
TONS	TDRS On-board Navigational System
TRMM	Tropical Rainfall Measuring Mission (joint US-Japan)

TSDIS	TRMM Science Data and Information System
USNO	US Naval Observatory
UT	universal time
UTC	universal time code
V0	Version 0
VIRS	Visible Infrared Scanner (TRMM)
WAIS	Wide Area Information Server
WKBCH	Workbench CSCI
WKSHW	Working Storage HWCI
WWW	World-Wide Web

This page intentionally left blank.