Planning Database System (PDS) is an **On-premises software** (sometimes abbreviated as "onprem") installed and run on computers on the premises which serves as a **platform for data**

storage. It **holds the data** which is being **sent across various systems**.

Data is brought in planning database from GEHC systems like **GLPROD, ITCS, SBOM, IB and MWS**

as source and is sent over to **Servigistics** for planning post two layers of data massaging i.e.

Business Rule and SPM Rule.

Data is brought in planning database from Servigistics and is sent over to GEHC systems like

**GLPROD, GLPROD FTP Locations, Informatica FTP** Locations post data massaging

Briefly, data will be captured in PDS and several business logics will be applied and thereafter

the transformed data will be fed to the destination system.

**Execution Tables:**

1. **GE\_PLN\_TRANSFORMATION\_TABLES**: Contains the Table **Name**, Table **Alias** and Table **Code**

2. **GE\_SPM\_RULE\_HEADERS\_ALL**: Contains all the **Rules Headers Details** which is mentioned in the set up document.

3. **GE\_SPM\_RULE\_LINES\_ALL**: Contains all the **Rules Lines Details** which is mentioned in the set up document.

4. **GE\_PLN\_TRANSFORMATION\_BASE\_DTL**: Contains all the **base queries for each activity.**

The base query will be determined by **activity type** and **logical flow**.

**Activity type** depicts the type of data modification activity like **insert** or **update**.

**Logical flow** determines the layer of flow of data i.e. **IP** (Dataflow from Inbound table to Processed table in PDS) and **PS** (Dataflow from Processed table to SPM table in PDS). All the base queries are mentioned in the set up document.

5. **GE\_PLN\_TRANSFORMATION\_EXEC\_DTL**: Contains all the **final validated query** **ready** for **execution**.

6. **GEMS\_IFACE\_SPM\_TABLE**: Contains the **execution details** of the **activity** .

7. **GEMS\_IFACE\_SPM\_TABLE\_DETAILS**: Contains every **steps of the execution details** for any **activity** .

8. **GE\_IFACE\_SPM\_RESTART\_MW**: Contains all the **activity names** with **unique process\_id** and **restart\_step\_id**. This table is solely used by **Boomi** to **restart an activity** in case of any failure situation.

The activity restart identification denotes the mode of failure:

a. Restart id **1** signifies **end to end flow needs** tobe **performed**

1. Restart id **2** signifies the **failure** at **PDS Stored Procedure**
2. Restart id **3** signifies the **failure** at **file transfer** from **PDS to destination system**

The Restart Id is maintained from Boomi when process starts. Also Boomi controls the id after completion of each individual steps and maintains the progress of the flow depending on this id.

1. **GEMS\_MW\_IFACE\_LOG\_IT\_TBL**: Contains **every steps of the execution details** for any **activity** from **Middle Ware** end.

**Packages :**

**1.** GE\_PLN\_**TRANSFORMATION:**

This package is responsible to

**create a query**

(It will build the executable queryconsidering the **base query from GE\_PLN\_TRANSFORMATION\_BASE\_DTL**)as per

the **user created rules**

( the user provided **conditions** **from** **GE\_SPM\_RULE\_LINES\_ALL** for a particular activity\_name, activity\_type, data\_stream and process\_flow )which will

be **executed at the time of data flow.**

And

**After** the **executable** is **built**, **GE\_PLN\_TRANSFORMATION.VALIDATE\_QUERY**  will **validate the query**.

After successful validation , the **executable query** is **saved in** **GE\_PLN\_TRANSFORMATION\_EXEC\_DTL** table.

The **error** will be **stored into** the table **GE\_RULE\_COMPILATION\_ERROR\_DTL** in case any failure happened during validation time .

1. **build\_query**: Builds the query **from** the **rules details** and **base query** using the tables

**GE\_SPM\_RULE\_HEADERS\_ALL**,

**GE\_SPM\_RULE\_LINES\_ALL**,

**GE\_PLN\_TRANSFORMATION\_TABLES** and

**GE\_PLN\_TRANSFORMATION\_BASE\_DTL**.

1. **validate\_query**: Validates if the **query** is **correct or not**.

Essentially validates the where clause. **If** there is any **validation error**, the same error **message** will be **inserted** **into** the table - **GE\_RULE\_COMPILATION\_ERROR\_DTL**

1. **save\_query**: **Saves** the **query** **into** the table **GE\_PLN\_TRANSFORMATION\_EXEC\_DTL**.

1. **execute\_transformation**: **Executes** the **saved query from** the table **GE\_PLN\_TRANSFORMATION\_EXEC\_DTL**.
2. **create\_query**:  This procedure performs the following sequential actions:
3. **Builds the query by calling the procedure build\_query.**
4. **Validate the query by calling the procedure validate\_query .**
5. **Save the query by calling the procedure save\_query**

* 1. GE\_PLN\_**TRANSFORMATION\_CALL**

GE\_PLN\_TRANSFORMATION\_CALL package **has** a **single procedure** **GE\_PLN\_TRANSFORMATION\_FLOW**.

This is driving implementation of 2 layers of logic i.e **Business logic** and **SPM logic**.

1. This procedure  allows user to enter the following **parameters** : **Activity Name**, **Data Stream** , **Activity Type** and the **flow** between layers.
2. The program will **accept** the **activity name** and will **check** the **following validations** for each :
3. It will check if this **program is already running for** the **given activity name within last 5 hours**. **If so**, it will **not allow** **rerunning this stored procedure**.

The stored procedure will **only run if** **prior** invoked **stored procedures** having the **same activity name** are **completed with status success or error** depending on the process flag.

1. This program will **throw** an **error** if the **activity name** is **NULL**

.

1. **After** the **successful validation** **data** will **flow** **from** Inbound (**INBD**) to Processed (**PRSD**) layer i.e. **IP** flow and then Processed (PRSD) to Outbound (SPM) layer of tables i.e. **PS flow**.
2. A **procedure** **GE\_PLN\_REQUEST\_SET** is **introduced** which will be **invoked** to **call** **child transformations** within any Parent transformations at below levels
3. Before-IP -> Prior to executing IP
4. Before PS -> After executing IP, Before executing PS
5. After-PS -> After executing PS
6. Before-Archive -> After executing PS, before executing Archive

The **sequence** and **control** will be defined in ‘**GE\_PLN\_FUNCTIONAL\_MAPPING\_TBL**’ Table where **MAPPING\_TYPE** is '**REQUESTSET**'.

1. GE\_**INBD\_PRSD\_STUB**

This stub is for the **execution** **of** the **logic** needs to be executed **before** the **records inserted into PRSD** table from INBD table and also **checks** the **mutual exclusiveness** of the **activity\_name** through lookup value in **GE\_PLN\_TRANSLATION\_LOOKUP** table

1. GE\_**PRSD\_SPM\_STUB**

This stub is for the execution of the logic needs to be executed before the records inserted into SPM table from PRSD table

1. GE\_**SPM\_STUB\_LOGIC**

 This stub is for the execution of the logic needs to be executed after the data inserted into the SPM table.

1. **GE\_SPM\_STUB**

This STUB is designed for implementing the archive logic.

1. **Critical to Process Variables**
   * 1. PROCESSED\_FLAG should be populated as ‘N’ means New in Inbound tables for data to be able to flow between INBD à PRSD à SPM Layer of tables.
2. **Definitions**

|  |  |
| --- | --- |
| SPM | Service Parts Management - Global planning tool |
| PDS | Planning Data Source |

1. Process flag of the tracking table : GEMS\_IFACE\_SPM\_TABLE holds the **status of the procedure**.
   * 1. The process flag denotes:
2. **‘I’**:  When the Store procedure is **initiated** the status flag will be set as ‘I’.
3. **‘C’**: When the Store procedure is completed successfully the status flag will be    set as ‘C’.
4. **‘W’**: When the store procedure is **completed with warning** then the status flag will be set as ‘W’.
5. **‘E’**: When the Store procedure is **completed  with error** the status flag will be set as ‘E’

A diagram of data processing

Description automatically generated

Above is achieved by following procedure/packages :

1.3.1 **GE\_PLN\_TRANSFORMATION Package**

GE\_PLN\_TRANSFORMATION package is responsible for **creating an executable query**.

The executable query for a **particular activity\_name**,  activity\_type, data\_stream and process\_flow is built **considering** the **details from base query** i.e. GE\_PLN\_TRANSFORMATION\_**BASE**\_DTL table **and** the **user provided conditions** from GE\_SPM\_**RULE**\_LINES\_ALL table. 

This program will **first check** the **active rule condition** in GE\_SPM\_**RULE**\_LINES\_ALL table corresponding to the **active rule** in GE\_SPM\_RULE\_**HEADERS**\_ALL table for a **particular activity name**. 

After the executable is built, this program **validates the query**. 

After successful validation the executable query is **saved** in GE\_PLN\_TRANSFORMATION\_**EXEC**\_DTL for a particular activity\_name, activity\_type, data\_stream and process\_flow. 

1.3.2 **GE\_PLN\_TRANSFORMATION\_CALL Package**

GE\_PLN\_TRANSFORMATION\_CALL package is **responsible** for data flow **from** **INBD** table of PDS to outbound (**SPM**) table of PDS.

It will first move the data from INBD table to PRSD table which is identified as **IP logical flow**. Then the data will move from PRSD table to outbound (SPM) table which is identified as **PS logical flow**. 

This program will take **Activity Name** as an **input** and will **consider** all the

**active rules** (where ENABLE\_FLAG is ‘**Y**’ for records in GE\_SPM\_RULE\_**HEADERS**\_ALL table) defined on the GE\_SPM\_RULE\_HEADERS\_ALL **for** the **input** **activity name**. 

These **active rules** will be **executed** in **order of priority** defined in the DATASTREAM\_**WEIGHTAGE** field of GE\_SPM\_RULE\_**HEADERS**\_ALL table. 

**For every active rules** in logical flow **IP**, the execution query for activity type ‘**INSERT’** will be **executed in** order of the **priority** set in the ACTIVITY\_SEQUENCE.

**After** this is successfully **completed**, the program will execute the execution query for activity type ‘**UPDATE’** **in order** of the **priority** set in the ACTIVITY\_**SEQUENCE** of logical flow **IP**. In this manner execution will be **performed for all active rules**.

**After** the successful completion of **IP flow**, all the **rules** having active **SPM enabled flag**  (SPM\_ENABLE\_FLAG is ‘**Y**’ of GE\_SPM\_RULE\_**HEADERS**\_ALL table) will be **considered** subsequently **for** the **PS** logical **flow**. 

Then **again** the execution query for the activity type ‘**INSERT**’ will be **executed** in **order** of the **priority** set in the ACTIVITY\_**SEQUENCE**. **After** successfully **completion** of **PS ‘INSERT’** execution query, the execution query for activity type ‘**UPDATE’** will be executed in order of the priority set in the ACTIVITY\_SEQUENCE. **After** successful **completion** of **PS** flow, **data** will move **to** the outbound **SPM** table from **PRSD** table.

There are four stubs which are triggered by this package.

1. **GE\_INBD\_PRSD\_STUB –**

This stub is executed **before** the commencement of **IP** flow. It **executes** the **logic** **of** all the intended **ammendments** on the records **in** the **INBD** table which are **inserted** **into** **PRSD** table **after checking** the mutual **exclusiveness**.

If the process gets initiated when any corresponding mutually exclusive process is already running, then the respective process will wait untill the mutually exclusive process gets completed.

1. **GE\_PRSD\_SPM\_STUB**:- This stub is executed **after** the **IP** flow and **before** the **PS** flow. Through this stub, records in PRSD table is  ammended.
2. **GE\_SPM\_STUB\_LOGIC**:- This stub is executed **after** the **PS** flow. Through this stub **SPM** **table** is **ammended**.
3. **GE\_SPM\_STUB**:- This STUB is designed mainly for implementing the archive logic and purge logic

Archival of a particular table and Purging is at the discretion of users and can be handled through the control table i.e. GE\_PLN\_SYSTEM\_CONTROLS

A procedure **GE\_PLN\_REQUEST\_SET** is introduced which will be invoked to call child transformations within any Parent transformations at below levels

i) Before-IP -> Prior to executing IP

ii) Before PS -> After executing IP, Before executing PS

iii) After-PS -> After executing PS

iv) Before-Archive -> After executing PS, before executing Archive

The sequence and control will be defined in ‘GE\_PLN\_FUNCTIONAL\_MAPPING\_TBL’ Table where MAPPING\_TYPE is 'REQUESTSET'. Below is how the mapping will be determined :

|  |  |
| --- | --- |
| **MAPPING\_TYPE** | REQUESTSET to determine the mapping for parent-child transformation relationships |
| **INTERNAL\_USE** | Parent Transformation |
| **ACTIVITY\_NAME** | Child Transformation |
| **ACTIVITY\_TYPE** | Control for the level of run : BEFORE\_IP, BEFORE\_IP, BEFORE\_PS, AFTER\_PS |
| **MAP\_VALUE1** | Frequency of child transformation runs |
| **CHAR\_VALUE1** | Enabled/Disabled |
| **NUMERIC\_VALUE1** | Priority |
| **CREATED\_BY** | SSO of the creator of this parent-child transformation relationships |
| **CREATION\_DATE** | Date of creation |
| **MODIFIED\_BY** | SSO of the modifier of this parent-child transformation relationships |
| **MODIFICATION\_DATE** | Date of modification |

A diagram of a company

Description automatically generated