Tanmoy Mondal
January 5, 2014 3 minute read

LUW – Part1

Logical unit of work (LUW) in SAP:

The objective of this document is aimed at explaining the concept of logical unit of work in SAP. I assume that this will be a good read for my fellow readers.

Why logical units of work?

The main purpose of the logical unit of work is to ensure data consistency in the R/3 systems. Let us consider that there is a complex R/3 transaction that involves a series of dialog steps; during the execution of one such dialog step at the middle of the transaction the data involved will lack completeness & will therefore be inconsistent. When talking about of logical unit of work we must note the following points initially.

1. Database LUW also known as DB LUW, each dialog step has its own DB LUW that is used to ensure data consistency.
2. SAP LUW consists of several dialog steps and all the changes to database are written in a single DB LUW. We must note that the database changes are always written in the final database LUW.
3. Therefore to summarize a complex R/3 transaction may span across several dialog steps each having its own DB LUW, however all the changes are written to the database in the final DB LUW of the SAP LUW after a commit work is executed by the system or rolled back with no changes written to the database due to errors.

What is a database LUW?

A database LUW is work unit consisting of database operations (INSERT, UPDATE, MODIFY & DELETE) that it used to make the database changes. The final database LUW of a SAP LUW is used to make the changes to the database, the changes to the database is not made until after the database commit. The database LUW is executed in a special kind of work process called the update work process.

What is a SAP LUW?

A SAP LUW is a logical unit work that spans over several dialog steps. Let us consider a complex R/3 transaction of n screens; where each screen has its own DB LUW involving database operations and are logically related to each other. It is important that the changes to the database for this complex transaction must be made all at once or it must be rolled back all for the data consistency; that is where the SAP LUW becomes critical in maintaining a logical relationship between several DB LUWs (dialog phases) & making all the changes to the database in the final DB LUW. SAP LUW uses the bundling technique to achieve the same. There are several possibilities of bundling technique and one of them is bundling the database changes using a function module call in the UPDATE TASK.

Bundling in SAP LUW:

A CALL FUNCTION...IN UPDATE TASK is one of the bundling techniques to achieve all the database changes in the last DB LUW of the SAP LUW. The open SQL statements for database changes are placed in the function module instead of placing them in the code, using the addition UPDATE TASK ensures that the function module is not executed until COMMIT WORK is found in the program. At COMMIT WORK the function module calls with UPDATE TASK are executed in a special work process called update work process and the changes to the database is executed in the final DB LUW of the SAP LUW.

R/3 transactions using update function modules.

Let us consider there is one R/3 transaction calling another R/3 transaction and as well as calling some update function module. There is some typical behavior that can be noted for this particular scenario.

- (a) The called transaction begins in a new SAP LUW in comparison to the calling transaction. It implies that there are two different SAP LUWs now.
- (b) A new update key is generated for the new SAP LUW that is used to identify all the update function modules.
- (c) If the called R/3 transaction does not have a COMMIT WORK then the update function modules will not be executed i.e. both the calling transaction & the called transaction must have their own COMMIT WORK.
- (d) The update function modules are called only after the system has executed a COMMIT WORK.
- (e) Let us consider there is an importing parameter -IM_X for the update function module; now the same function module is called at several places in the program like shown below –

```
DATA: gv_x TYPE i.

gv_x = 1.

CALL FUNCTION UPD_FUNC IN UPDATE TASK

EXPORTING

    IM_X = gv_x.
```

```
gv_x = 2.

CALL FUNCTION UPD_FUNC IN UPDATE TASK

EXPORTING

    IM_X = gv_x.

gv_x = 3.

COMMIT WORK.
```

What will be the value of gv_x at the COMMIT WORK?

No, the value of gv_x will not be 3; when commit is executed by the system the value of gv_x that will be passed is dependent at the point of function call therefore during the first call the value of gv_x will be 1 & during the second call the value of gv_x will be 2.

Assigned tags

ABAP Development |

Related Blog Posts

SAP Archiving Activity – Part1
By **Former Member** , Sep 13, 2013

A small tip of class cl_system_transaction_state
By **Jerry Wang** , Jan 03, 2014

How to perform a simple BDC – Part1
By **Former Member** , Oct 30, 2006

Related Questions

Is there a way to get ABAP Stack Information of a previous LUW?
By **Mou Bhattacharya** , Feb 02, 2018

Allow Within an LUW (Transaction_Manager) COMMIT WORK AND WAIT
By **Former Member** , Mar 08, 2018

Difference between LUW and Parallel Processing
By **Vivek Khobragade** , May 05, 2019

6 Comments

You must be **Logged on** to comment or reply to a post.

Former Member
January 6, 2014 at 8:17 am

Hi Tanmoy,

Also , the system does an implicit commit after every dialog step, which might result in data inconsistency. Using update FM avoids that problem.

Br,

Shankar.

Like (0)

Suhas Saha

January 8, 2014 at 9:50 am

> *Also , the system does an implicit commit after every dialog step, which might result in data inconsistency. Using update FM avoids that problem.*

An implicit database commit is triggered at the end of every dialog step – there is a difference between COMMIT WORK & Db commit.

– Suhas

Like (0)

Sijin Chandran

January 6, 2014 at 11:28 am

Yeah this is really a good read 🙂

Like (0)

Suhas Saha

January 8, 2014 at 9:48 am

Hello Tanmay,

Imho this content is redundant & already discussed multiple times. But it's nicely written 🙂

Next time please use your judgement if the content is redundant & basic. You wouldn't like your content to be rejected on that ground, would you?

Keep up the good work.

BR,

Suhas

Like (0)

Tanmoy Mondal | Post author

January 11, 2014 at 3:27 pm

Ok Suhas I will keep this in mind

BR

Tanmoy

Like (0)

Galen Schliem

March 22, 2017 at 10:59 pm

Very helpful. Thanks Tanmoy!

Galen

Share & Follow