

# Unit 6 – Connecting IMS to CICS and DB2

What this unit is about

External subsystems might need to connect to IMS for several important reasons. CICS systems might need to connect to an IMS Control Region so that CICS transactions can have access to IMS DB data. IMS transactions might need to connect to DB2 subsystems in order for access to data stored in DB2.

What you should be able to do

After completing this unit, you should be able to:

Learn how IMS can serve as a database manager for CICS transactions

Describe the steps required to allow access between IMS and DB2 and between IMS and CICS

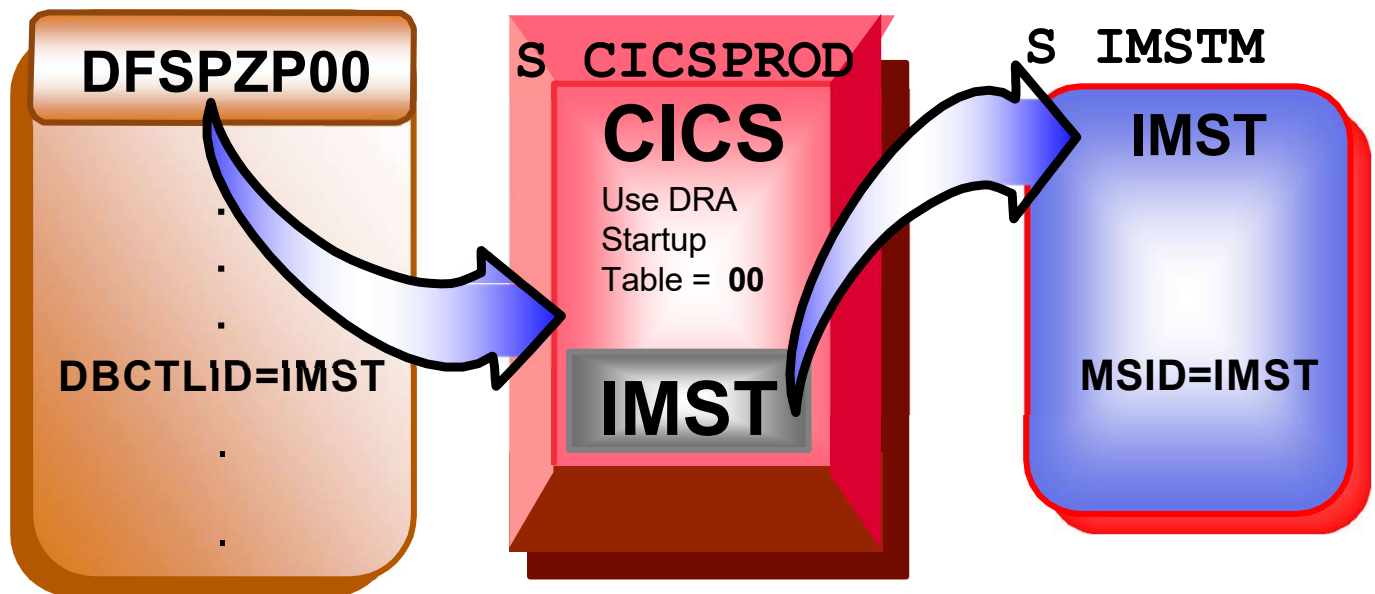
Understand the role of the transaction manager and the database manager in the two-phase commit process

Describe how the two-phase commit works between CICS and IMS and between IMS and DB2 in order to assure data consistency across database managers

Attention : also newest WAS connectivity locally will use ESAF !

# CICS Connection to IMS (1 of 2)

CICS Transactions can use IMS as an External Database Manager



## Notes:

A CICS Connection is always initiated from CICS either at CICS startup or from CICS terminal via a dialog by using a specific Database Resource Adapter (DRA) startup table.

A CICS can only be connected to one IMS at a time. CICS can disconnect from IMS and connect to another IMS by using another DRA startup table without restarting CICS.

We're talking about DLI ...☺ - NOT the CICS-IMS connection as an ISC (VTAM!) !

TABLE = 00 ... or another DFSPZPxx suffix

# CICS Connection to IMS (2 of 2)

- When CICS is started, the connection to IMS can be restarted automatically by including DFHDBCON in the PLTPI list
- CICS and IMS can be started and stopped independently:
  - The connection between CICS and IMS is made by using the CICS transaction

C D B C

- This transaction invokes program DFHDBCON
  - A parameter in the DRA startup table passed to this program identifies which IMS system that this CICS should connect to
  - If IMS is not available the transaction CDBC can retry the request automatically after a user-specified interval (TIME= parameter in DRA startup table)
- At CICS Warm Start / Emergency Restart, the DRA module with the suffix last used will be loaded.
  - At Cold Start, the DRA module with suffix 00 will be used automatically

KC110 unit 6 page 3

## Notes:

There is a transaction provided by CICS that enables the stopping and starting of the connection between CICS and IMS.

# Enabling CICS - IMS connection (1 of 2)

- The DRA module with suffix other than 00 can be used automatically by specifying:

**INITPARM=(DFHDBCON='xx')** in the SIT

- The connection can be terminated using the CDBC transaction
- CICS Resource Definition:
  - Include group DFHDBCTL for IMS required resources
  - Destination Control Table (DCT)

**DFHDCT: DESTID=CDBC**

- Program List Table (PLT) (post initialization) for automatic connect:

**DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM DFHPLT  
TYPE=ENTRY,PROGRAM=DFHDBCON**

- Use CICS Tran CDBI to display CICS-IMS Connection Status

# Enabling CICS - IMS connection (2 of 2)

- IMS operator commands from CICS:
  - Can be issued using the CICS supplied transaction CDBM

```
DFHDCT: DESTID=CDBM
```

- IMS Steps required to enable the CICS to IMS interface:
  1. Generate a PSB (and corresponding ACB) with no PCBs: PSBGEN

```
LANG=ASSEM,PSBNAME=DFHDBMP,IOASIZE=1000
```

1. Define to IMS with an APPLCTN macro (or DRD equivalent):

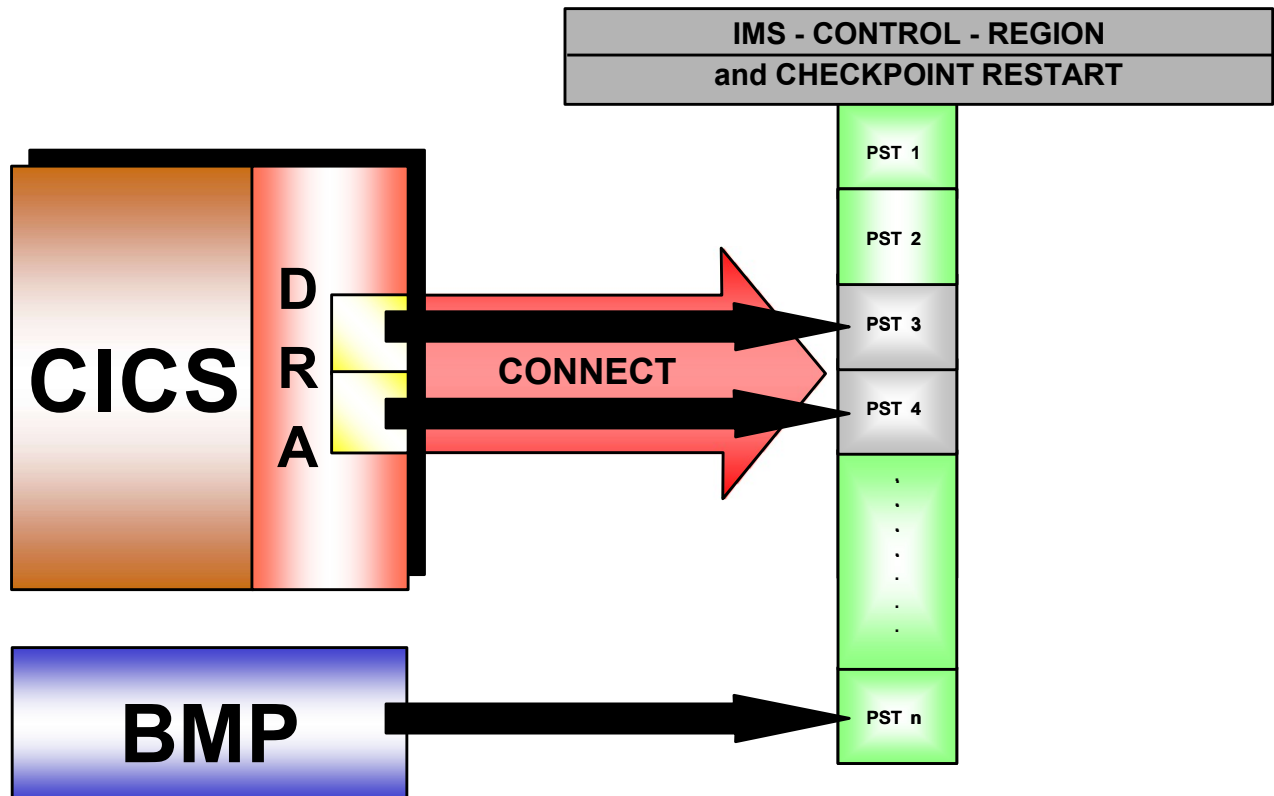
```
APPLCTNPSB=DFHDBMP,SCHDTYP=PARALLEL,PGMTYPE=BATCH
```

## Notes:

Not all IMS commands can be issued from CICS CDBM.

Details of the CICS to IMS attach are described in the CICS IMS Database Control Guide CICS manual.

# CICS / IMS Interface CB Structure



The Database Resource Adapter (DRA) modules and DRA startup table are loaded. The DRA is:


- IMS code
- Resident in CICS address space
- The interface between CICS and IMS

Thread TCB attached and threads established dependent on MINTHRD= parameter in DRA startup table. A THREAD is:

- ✓ A set of control blocks which provide a communication path between a CICS transaction and IMS.
- ✓ Maximum of 999 concurrent threads.
- ✓ z/OS TCB associated with each thread.
- MINTHRDs allocated at start of connection.
- Additional threads can be created dynamically up to a maximum of MAXTHRD= or MAXPST=.
- Each thread is assigned to a specific PST.
- PSTs assigned to MINTHRD= threads are never freed as long as the CICS-IMS connection exists.
- PSTs assigned to additionally created threads (up to MAXTHRD= or MAXPST=) are freed as soon as the thread is *collapsed*. This is expensive from a CPU perspective since it requires terminating the z/OS TCB.

# CICS PSB scheduling

- At the time the CICS–IMS Connection is established, CICS pre builds *MINTHRD* threads and the associate control blocks
  - Including z/OS TCBs ; remain after terminate
- Later, a CICS transaction requests a thread by issuing the CICS DLI *Schedule PSB xxxxxxxx* request
  - If free thread is available, it is assigned
  - If no free thread and current number of threads less than MAXTHRD (and number of IMS PSTs less than MAXPST)
    - New thread created and assigned
    - This thread, and its associated TCB, are freed at the time this CICS task terminates
  - If no free thread and we are at limits described above, wait for thread
- When thread is eventually assigned, continue the IMS schedule process described in Transaction Schedule unit
  - Acquire pool space and load corresponding Scheduler control blocks
- Release IMS pool space and CICS Thread at Task Commit
  - Every CICS task instance requires the services of the IMS Scheduler

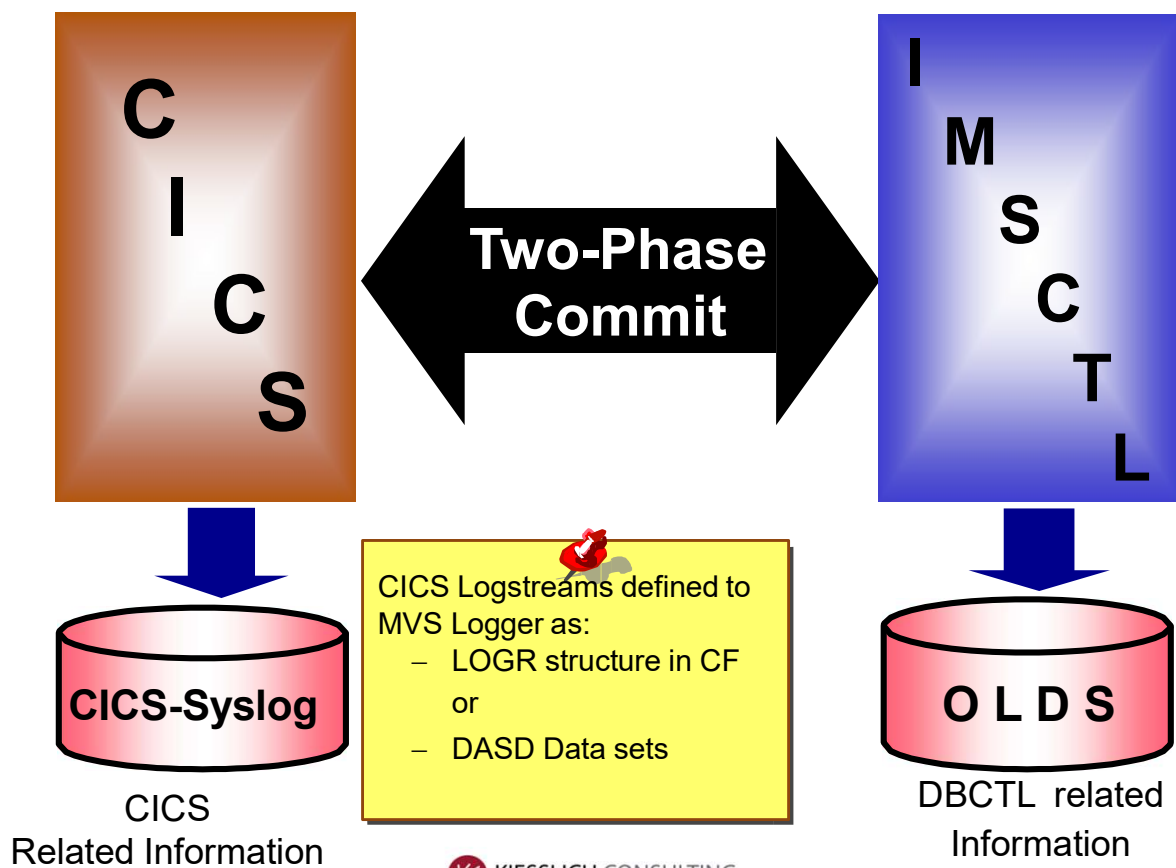
 IBM RESOURCES

KC110 unit 6 page 7

## Notes:

For CICS transactions that need to process DL/I DB calls, a special CICS Schedule PSB call is required; a scheduling request might be delayed if there are insufficient available CICS-IMS threads available. When the schedule PSB request is received by IMS, the IMS scheduler acquires storage and loads the appropriate scheduler related control blocks (PSB, DMB, and so on).

# CICS-IMS two-phase commit overview



KC110 unit 6 page 8

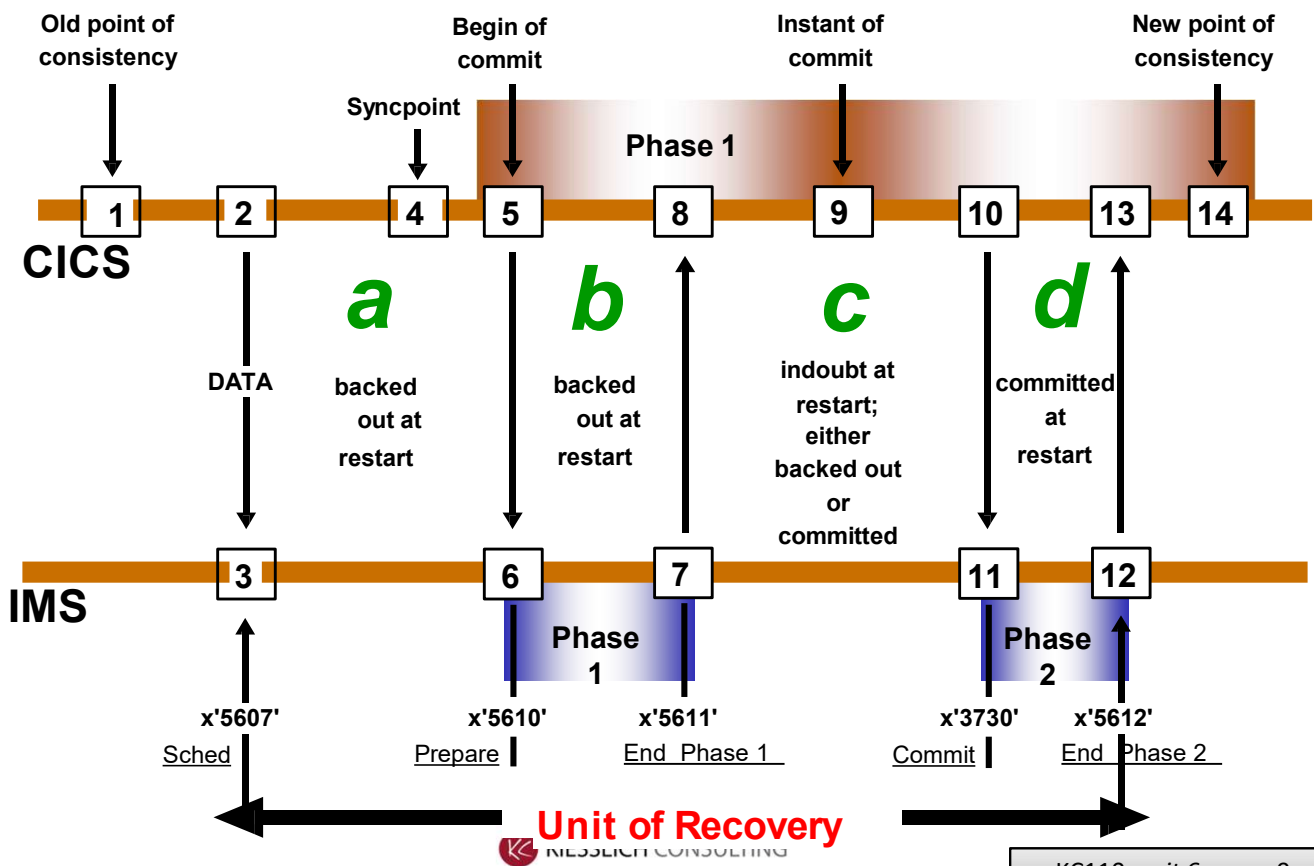
## Notes:

CICS is the Transaction Manager and Coordinator Controller (CCTL) that determines whether a UOW (and all work performed by participants in the UOW) should be COMMITTED or ABORTed (Backed out).

IMS is the database manager and Participant



# CICS-IMS two-phase commit process



## Notes:

The various processing steps of the two-phase commit are described in detail on the next several pages.

# CICS-IMS 2PC Process (2 of 4)

- A two-phase commit process is required when the IMS database system is accessed from CICS. To synchronize data between the database system and a transaction manager, any data change in one system must be matched by complete data changes in the other. Before either subsystem completes the commits of changed data, it must be decided whether all subsystem can make its corresponding change. The Change Coordinator (CCTL), CICS in this case makes the Commit or Abort decision. Throughout the two-phase commit, the subsystems must be able to communicate about the disposition of the unit of work.
- CICS and IMS use a two-phase commit process to communicate with each other. Extensive logging is performed by each subsystem to document its progress through the commit process. This logging is used to determine how to proceed (commit or backout) if a failure occurs during commit.
  - Phase-1 is initiated by the Change Coordinator (CICS in this case). It (the CCTL) instructs each subsystem to determine independently whether it has recorded sufficient recovery information in its log, and can commit its work. For Full Function databases this means changed data will be written to the database after the corresponding log records have been written. Phase-1 is also called *PREPARE PHASE*. At the end of Phase-1, the change coordinator (CICS in this case), determines whether it and all participants (IMS in this case) are able to proceed with phase-2 of the commit. If it decides that commit should proceed, it instructs all participants to begin Phase-2. Otherwise, it (CICS) tells all participants to Abort/backout this unit of work.
  - In Phase-2, held resources are freed (i.e. locks are released). Changed data is now available for other users). Phase-2 is also called *COMMIT PHASE*. Even if one subsystem/participant terminates abnormally during Phase-2, the operation is considered "complete" by the non-terminating systems/participants. Any remaining changes (mainly log updates) are applied and locks are released by the failed system the next time it restarts.
- The commit processing is initiated by an application syncpoint or PSB termination or normal CICS end of task processing (4).

# CICS-IMS 2PC Process (3 of 4)

- Phase-1 of commit processing begins (5).
  - As CICS begins the Phase-1 processing, so does the IMS (6). IMS successfully completes Phase-1 (7), writes this fact to the log, and notifies CICS. CICS receives the notification (8). If CICS determines that all participants (including itself) can commit, it logs that fact (9).
  - Now that the decision has been made, it will be honored after any subsequent failure and recovery of either system.
- CICS begins Phase-2 of the processing - the actual commit of the UOW:
  - It notifies IMS to begin its Phase-2 (10). IMS logs the start of Phase-2 (11) and completes it successfully at (12), which is then a new point of consistency for IMS.
  - CICS finishes its Phase-2 processing (14). The data controlled by both subsystems is now consistent and available to other applications.
- If either participant fails prior to the end of Phase-1, it does not need to be able to communicate with other subsystems in order to decide how to treat an inflight UOW –inflight UOWs will always be backed out during restart. Also, if either participant fails after starting Phase-2 starts, no further communication with the other subsystem is required either for it to free \*its\* resources. Only if a system fails in the indoubt window between Phase-1 end and Phase-2 start will it be required to reestablished communications in order to determine how to proceed.



# CICS-IMS 2PC Process (4 of 4)

The status of a unit of recovery after a failure depends upon the moment of failure.

- **INFLIGHT**

If CICS or IMS fail before finishing Phase-1 (period a or b); during restart, IMS backs out the updates.

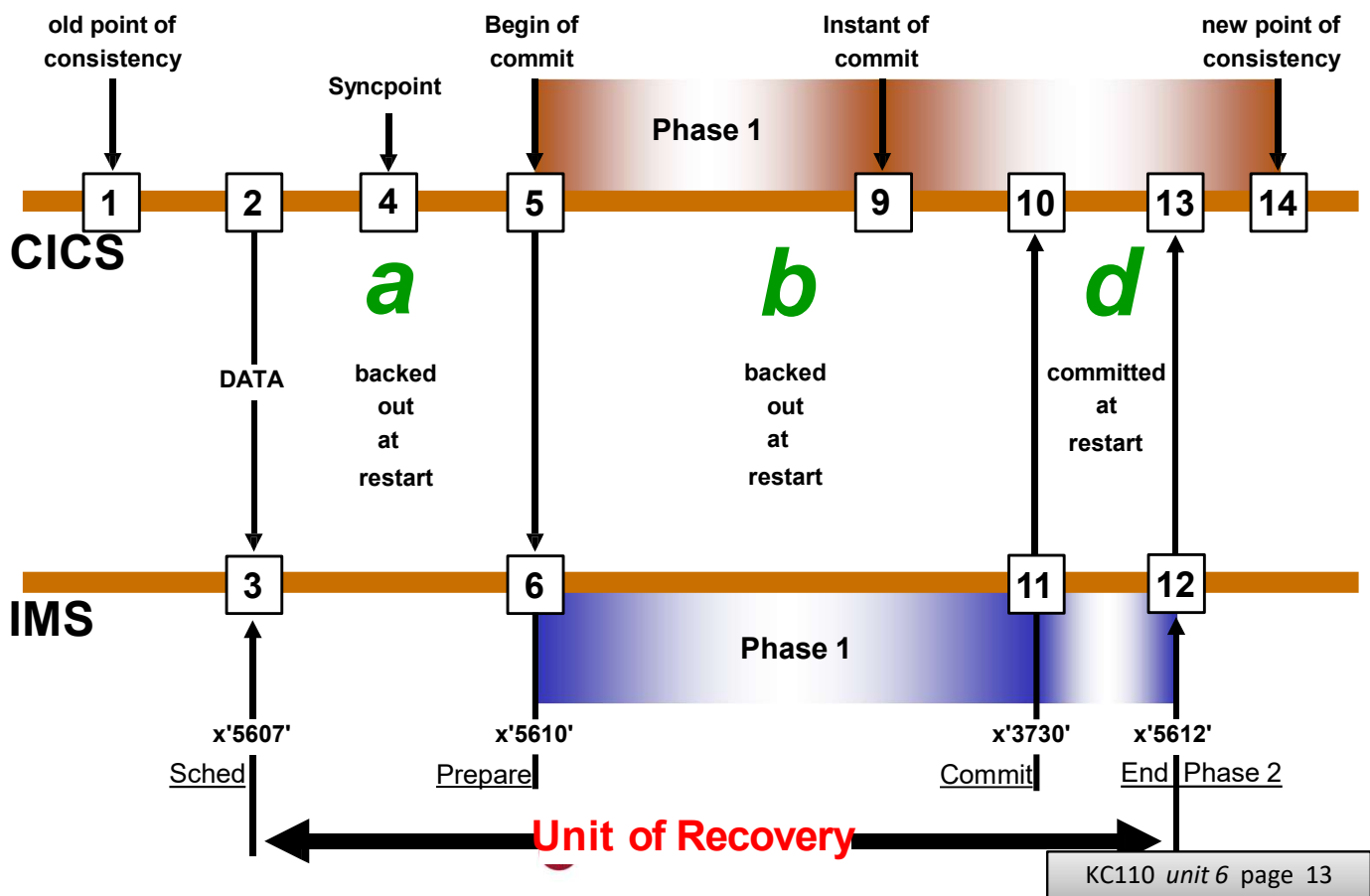
- **INDOUBT**

If IMS or CICS failed after finishing Phase-1 and before Phase-2 (period c); only CICS knows if the failure happened before or after the commit (point 9). If it happened before the commit it must be backed out; if it happened after, IMS must make the changes and commit them. If IMS failed, at restart, IMS waits for information from CICS before processing the unit of recovery. If CICS failed, IMS will patiently wait for instruction from CICS on how to proceed; CICS will inform IMS as part of its (CICS's) restart, what to do with all indoubt UOWs.

- **IN-COMMIT**

If IMS failed after it began its own Phase-2 processing (period d); it makes the committed changes as part of restart.

# CICS-IMS single-cycle commit process



## Notes:

If IMS is the only external resource manager for a specific UOR; even a different instance of the same transaction could involve changes involving a differing number of participants: zero participant, one participant (can use single cycle or two-phase commit), or multiple participants (and MUST use two-phase commit). The Commit Coordinator (CICS in this example) determines what to require of the participant(s) at application commit time.

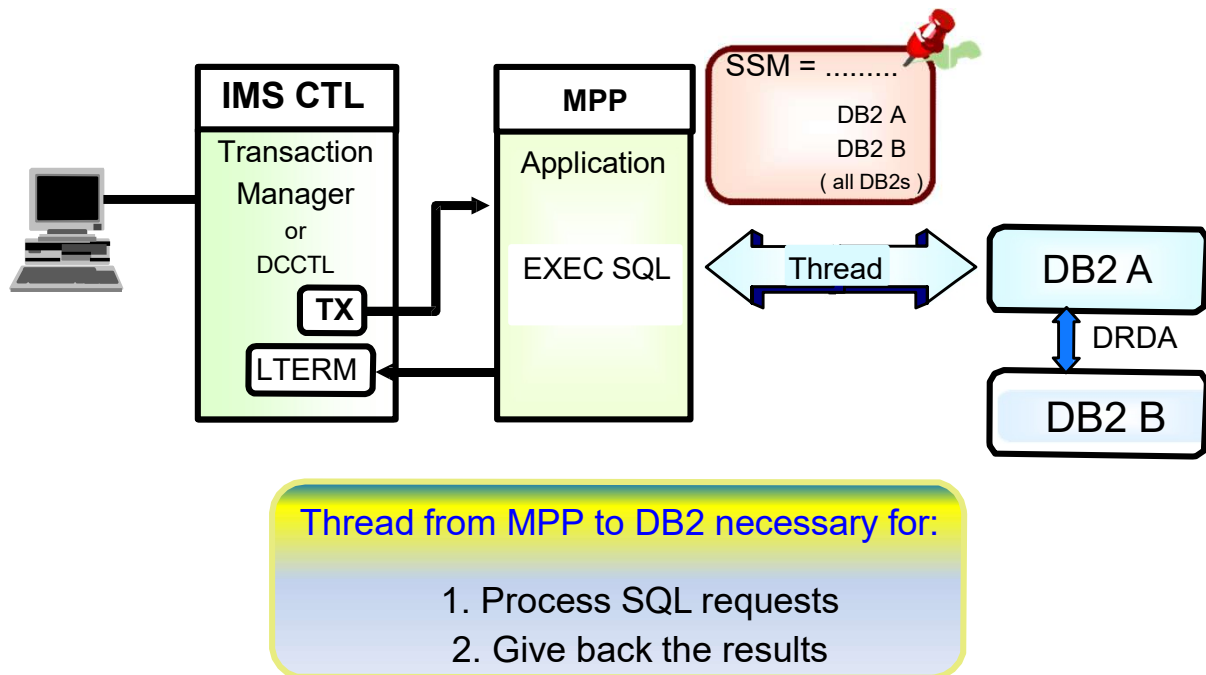
With single-cycle commit, CICS calls IMS once to commit or abort. Several benefits of SCC:

- Reduces number of log records
- \*End-Phase-1 (x'5611') and CHKW eliminated
- Reduces calls across interface
- No calls resulting from a PREPARE phase

No indoubt phase; the only reason that there is an *indoubt phase* with two-phase commit is that the Commit Coordinator needs to receive confirmation from multiple participants before deciding to commit

# IMS/TM connections to DB2 (1 of 2)

## IMS Transactions can use DB2 as an External Database Manager



### Notes:

Like the CICS to IMS connections, the connections between IMS and DB2 are called *threads*.

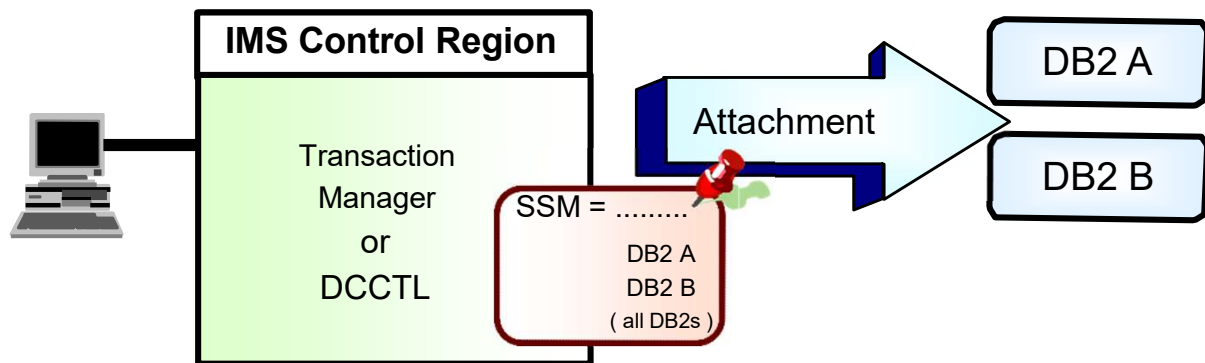
There are two types of IMS-DB2 threads:

- *Transaction threads* (shown on this page) exist between a dependent region and DB2 and are used to process SQL calls; there is a single transaction thread per IMS Dependent region.
- *Command threads* (shown on the next page) are established between the IMS Control Region and DB2, and are used to process commands and to coordinate two-phase (or single-cycle) commit processing.

In spite of the fact that there are two types of threads associated with the IMS-DB2 interface; from a systems programming perspective, the establishing and supporting of this interface is much simpler than the CICS-IMS interface.

EXT SUBSYS also : MQ , if application does specific MQPUT and MQGET calls !  
These APPLs are often triggered by MQ LISTENER BMP

# IMS/TM connections to DB2 (2 of 2)




*f* Attachment from CTL to DB2 necessary for:

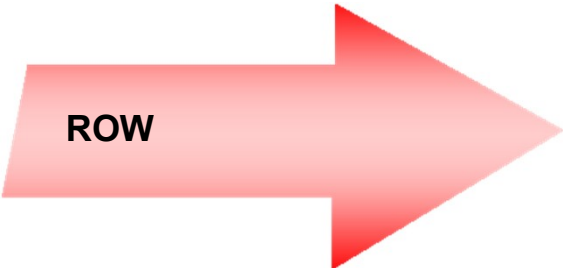
1. The two-phase commit (serializing of Sync Point Processing) or Single Cycle Commit
2. Commands from IMS to DB2

# What is a Relational Database?

- A *Relational Database* is perceived externally as a collection of *Tables*



DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
A00	COMPUTING SERVICE	000010	
B01	PLANNING	000020	A00



B01	PLANNING	000020	A00
-----	----------	--------	-----

- All data relationships are represented by column (field) values
- DB2 Tables are defined and manipulated using the SQL Language

## Notes:

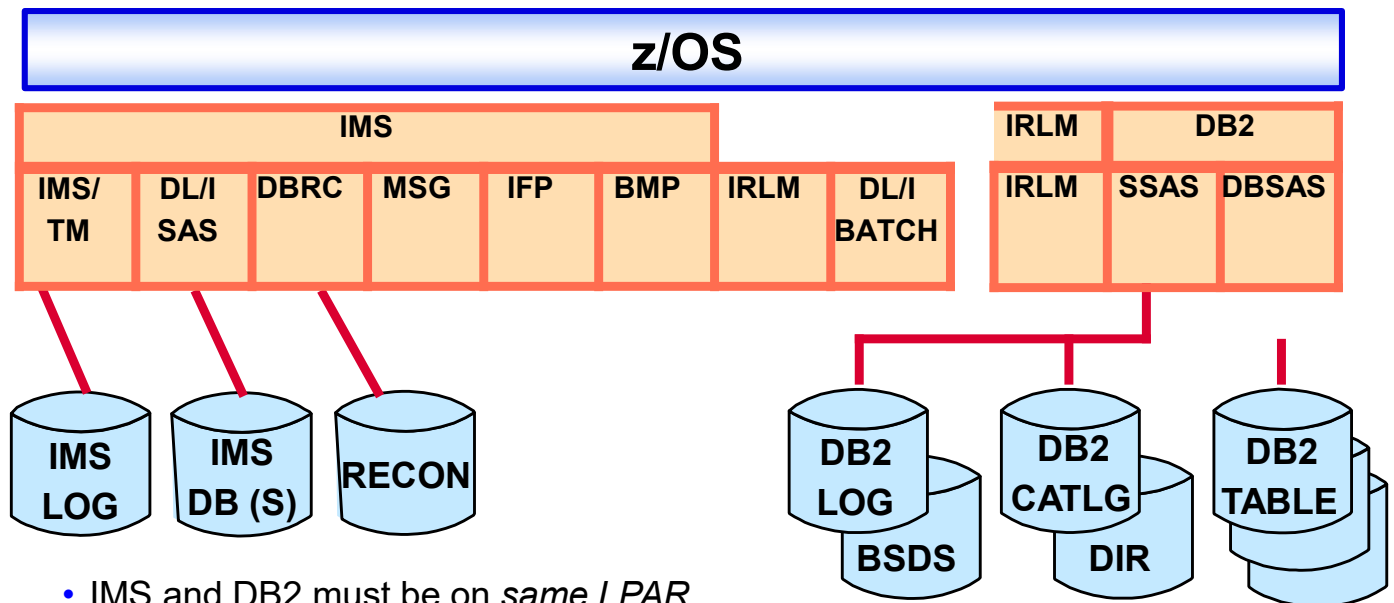
DB2 is the IBM relational database manager that runs on z/OS. Terminology for DB2 databases is different than the terminology associated with IMS databases. DB2 *tables* are analogous to IMS Segment types; DB2 *rows* correspond to IMS segment instances; DB2 *columns* correspond to IMS segment fields.



# IMS Attachment Facility (ESAF)

- Connection between IMS Subsystems and one or more DB2 Subsystems
  - A single dependent region can only be connected to a single DB2 system at a time
- Communication between IMS users and DB2
  - Command Thread
  - Transaction Threads
- Application Programming Interface:
  - DB2 Precompiler
  - Language Interface
- Coordinated Recovery/Restart
  - Unit of Work/Unit of Recovery
  - Two-Phase Commit Protocol
  - Indoubt Thread Resolution

# z/OS Environment



- IMS and DB2 must be on same LPAR
- One DB2 S/S can be connected to multiple IMSs
- One IMS S/S can connect to multiple DB2s
- One IMS dependent region (or batch IMS Job) can connect to a single DB2 at a time
- One application program can access only one DB2 per schedule

## Notes:

DB2 has a multi-address space architecture similar to IMS's.

Same LPAR : Xross Memory !

# IMS DB2-related commands

- **Subsystem commands processed by IMS:**

/DISPLAY SUBSYS xxxx

Displays status of connections

/STOP SUBSYS xxxx

Disconnects DB2 from IMS

/START SUBSYS xxxx

Reconnects DB2 to IMS after /STOP

- **Subsystem Request command forwarded to DB2:**

/SSR -Display ....

DB2 Command “-DISPLAY ....” sent to DB2

## Notes:

These are some of the IMS commands that can be issued from IMS that can control DB2. The functioning of these commands will require appropriate security authority. Note the “-” that precedes the DISPLAY command. Just as IMS frequently uses “/” as a control character, “-” is a frequent subsystem recognition character for DB2.

# IMS-DB2 system definition

- No specific IMSGEN or DB2 installation requirements, except defining programs and transactions to IMS:  
APPLCTN PSB=PRGMA  
TRANSACTION CODE=TRANA
- Must define the IMS-DB2 connections
  - SSM member in IMS.PROCLIB define DB2 subsystem connections:
    - Easy to code – mainly needs DB2 subsystem name and subsystem recognition character
    - Different IMS regions might have unique members
- DB2 Precompiler:
  - Checks syntax of SQL calls
  - Replaces EXEC SQL with CALL DSNHLI
  - Creates DB Request Module (DBRM)
  - Includes timestamp in source and DBRM
- Compile *Modified* Source  
CALL DSNHLI
- Link-Edit with IMS call interface module (DFSLI000)  
Entry point is DSNHLI



KIESSLICH CONSULTING

KC110 unit 6 page 20

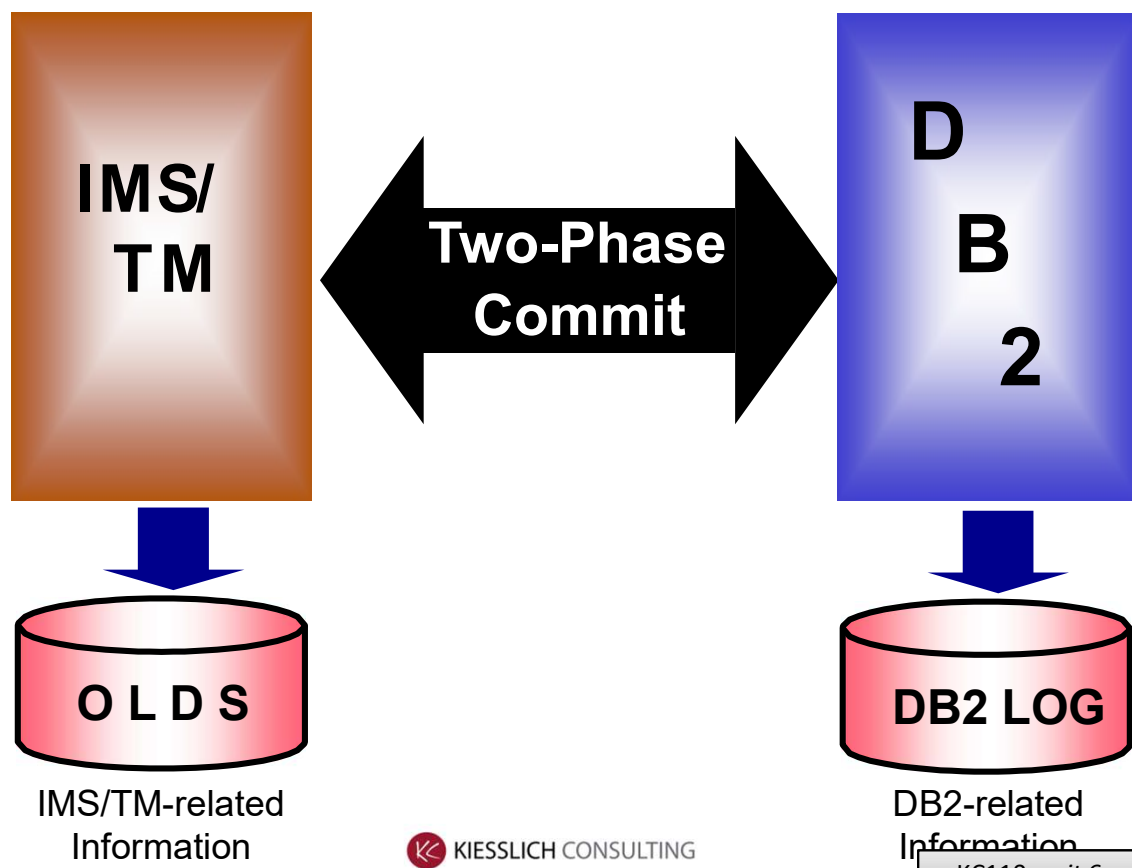
There is no indication in the IMS Gen (or DRD) definition for a program or transaction entry that specifies whether or not DB2 will be used; this obviously reduces IMS systems programmer involvement when application decide to use DB2.

The SSM names are based on IMSID concatenated to the SSM= value specified for IMS regions startup parameters. For example, IMS system IMSP with SSM=DB20 would result in member IMSPDB20 of IMS.PROCLIB being used. That member might contain multiple lines, each corresponding to a different DB2 system that this IMS is to connect to.

Applications that use DB2 (or CICS) must run a DB2-supplied pre-compiler step as part of the compile (or assemble) and link-edit job; there is no IMS pre-compiler.

ACBNAME = PLANNAME .. Or RTT definition ( Resource Translation Table) to relate ACBNAME to different PLANNAME

# IMS - DB2 two-phase commit overview

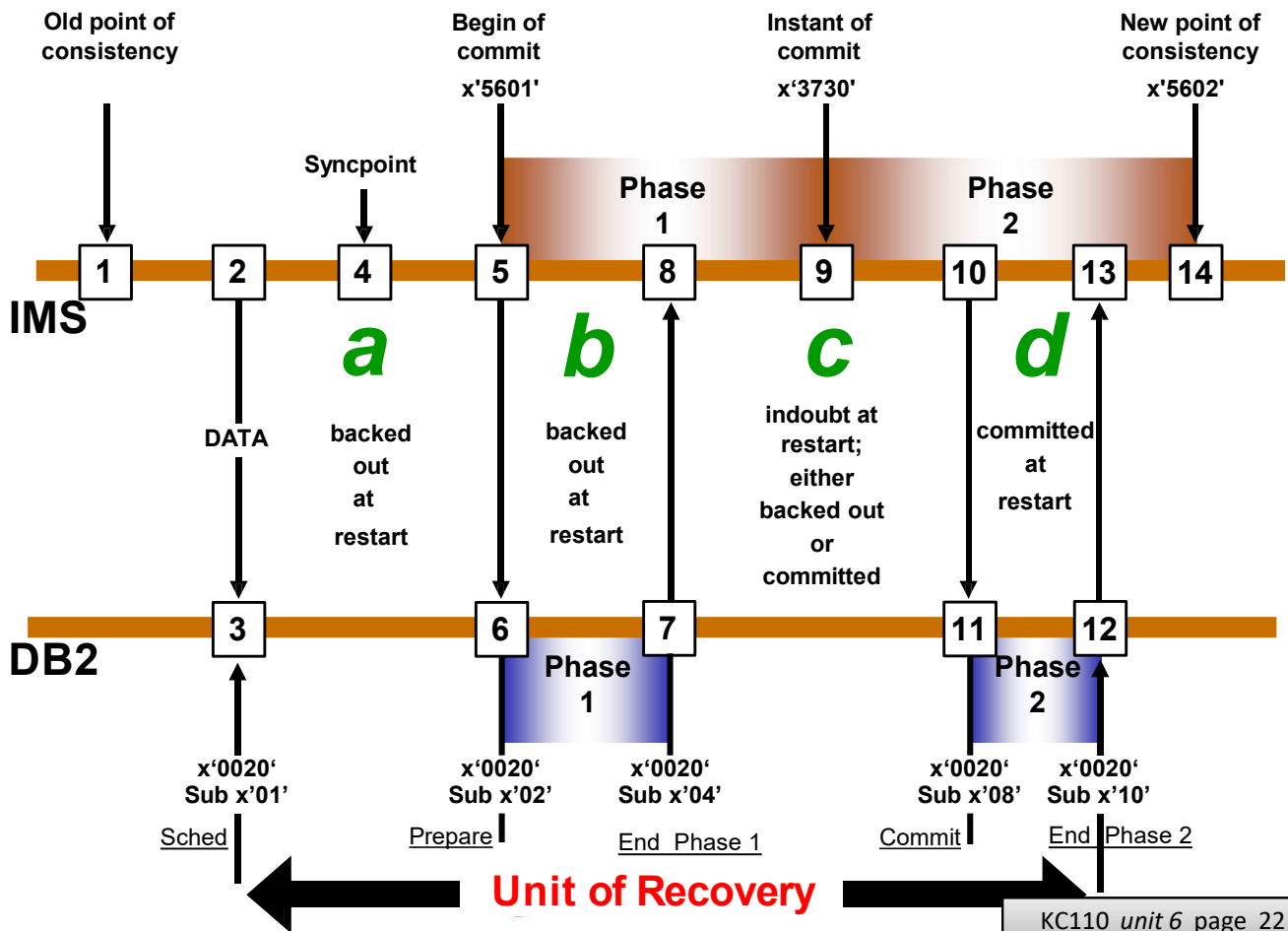


KC110 unit 6 page 21

## Notes:

Here IMS/TM is the Transaction Manager and Coordinator Controller (CCTL) that determines whether a UOW should be COMMITTED or ABORTed (Backed out). DB2 is the database manager and Participant

# IMS-DB2 2PC Process



Although not shown, the IMS-DB2 attach also supports single-cycle commit.

# IMS-DB2 2PC Process (2 of 4)

- A two-phase commit process is required when the DB2 database system is accessed from IMS transactions.

To synchronize data between the database system and a transaction manager, any data change in one system must be matched by complete data changes in the other. Before either subsystem completes the commits of changed data, it must be decided whether all subsystem can make its corresponding change. The Change Coordinator (CCTL), IMS in this case makes the Commit or Abort decision. Throughout the two-phase commit, the subsystems must be able to communicate about the disposition of the unit of work

- IMS and DB2 use a two-phase commit process to communicate with each other. Extensive logging is performed by each subsystem to document its progress through the commit process. This logging is used to determine how to proceed (commit or backout) if a failure occurs during commit.
- The commit processing is initiated by an application syncpoint or transaction termination process, GU to I/O PCB or MPR processing end (4).

Phase-1 is initiated by the Change Coordinator (IMS/TM in this case). It (the CCTL) instructs each subsystem to determine independently whether it has recorded sufficient recovery information in its log, and can commit its work. For DB2 databases this means that log records that correspond to changed data will need to be written to the DB2 log. Phase-1 is also called PREPARE PHASE. At the end of Phase-1, the change coordinator (IMS/TM here), determines whether it (and its changed IMS DB data) and all participants (DB2 here) are able to proceed with Phase-2 of the commit. If it decides that commit should proceed, it instructs all participants to begin Phase-2. Otherwise, it (IMS) tells all participants to Abort/backout this unit of work.

In Phase-2, held resources are freed (that is, locks are released). Changed data is now available for other users). Phase-2 is also called COMMIT PHASE. Even if one subsystem/participant terminates abnormally during Phase-2, the operation is considered complete by the non-terminating systems/participants. Any remaining changes (mainly log updates) are applied and locks are released by the failed system the next time it restarts.

# IMS-DB2 2PC Process (3 of 4)

- Phase-1 of commit processing begins (5):
  - As IMS/TM begins the Phase-1 processing, so does the DB2 (and IMS/DB) (6). DB2 successfully completes Phase-1 (7), writes this fact to the log, and notifies IMS/TM. IMS/TM receives the notification (8). If IMS/TM determines that all participants (including itself) can commit, it logs that fact (9).
  - Now that the decision has been made, it will be honored after any subsequent failure and recovery of either system.
- IMS/TM begins Phase-2 of the processing - the actual commit of the UOW:
  - It notifies DB2 to begin its Phase-2 (10). DB2 logs the start of Phase-2 (11) and completes it successfully at (12), which is then a new point of consistency for DB2.
  - IMS/TM finishes its Phase-2 processing (14). The data controlled by both subsystems is now consistent and available to other applications.
- If either participant fails prior to the end of Phase-1, it does not need to be able to communicate with other subsystems in order to decide how to treat an inflight UOW
  - inflight UOWs will always be backed out during restart. Also, if either participant fails after starting Phase-2 starts, no further communication with the other subsystem is required either for it to free *its* resources. Only if a system fails in the indoubt window between Phase-1 end and Phase-2 start will it be required to reestablished communications in order to determine how to proceed.





# IMS-DB2 2PC Process (4 of 4)

The status of a unit of recovery after a failure depends upon the moment of failure.

- **INFLIGHT**

If IMS/TM or DB2 fail before finishing Phase-1 (period a or b); during restart, DB2 (and IMS DB) backs out the updates.

- **INDOUBT**

If IMS/TM or DB2 failed after finishing Phase-1 and before Phase-2 (period c); only IMS/TM knows if the failure happened before or after the commit (point 9). If it happened before the commit it must be backed out; if it happened after, DB2 must make the changes and commit them. If DB2 failed, at restart, DB2 waits for information from IMS/TM before processing the unit of recovery. If IMS/TM failed, DB2 will patiently wait for instruction from IMS/TM on how to proceed; IMS/TM will inform DB2 as part of its (IMS/TM's) restart, what to do with all indoubt UOWs.

- **IN-COMMIT**

If DB2 failed after it began its own Phase-2 processing (period d); it makes the committed changes as part of restart.

# IMS RRS Attachment Facility (RRSAF)

- Connection between IMS Subsystems, DB2 Subsystems and maybe others (WAS)
  - A single dependent region can only be connected to a single DB2 system at a time
- Communication between IMS users and DB2
  - Command Thread
  - Transaction Threads
- Application Programming Interface:
  - DB2 Precompiler
  - Language Interface
- Coordinated Recovery/Restart
  - Unit of Work/Unit of Recovery
  - Two-Phase Commit Protocol
  - Indoubt Thread Resolution

## News

Watch out for recent changes in WAS / WOLA and IMS attach via ESAF !!!

( IMS V13 or V14 news )

# IMS-DB2 system definition

- No specific MSGEN or DB2 installation requirements, except defining programs and transactions to IMS:  
APPLCTN PSB=PRGMA  
TRANSACTION CODE=TRANA
- Must define the IMS-DB2 connections
  - SSM member in IMS.PROCLIB define DB2 subsystem connections:
    - Easy to code – mainly needs DB2 subsystem name and subsystem recognition character
    - Different IMS regions might have unique members
- DB2 Precompiler:
  - Checks syntax of SQL calls
  - Replaces EXEC SQL with CALL DSNRLI
  - Creates DB Request Module (DBRM)
  - Includes timestamp in source and DBRM
- Compile *Modified* Source and Link-Edit with IMS call interface module (DFSLI000)  
Entry point is DSNRLI

Applications that use DB2 (or CICS) must run a DB2-supplied pre-compiler step as part of the compile (or assemble) and link-edit job; there is no IMS pre-compiler.

ACBNAME = PLANNAME ...

Or use RTT definition ( Resource Translation Table) to relate ACBNAME to different PLANNAME