Introduction

The Adabas Triggers and Stored Procedures Facility is used to define, process, and monitor triggers and stored procedures.

This chapter introduces both types of procedures and their characteristics. It introduces the components of the facility and describes how they function together to provide online services and background processing.

This chapter covers the following topics:

- Procedures
- Components
- Processing Summary

Procedures

A *procedure* is a Natural subprogram that is written and tested using standard Natural facilities. The primary difference between triggers and stored procedures is the way they execute:

- A *trigger* executes ("fires") automatically when a specified event occurs-usually a data access or update to the associated table. The event occurs when selection criteria are satisfied. Selection criteria include file number and possibly command type or the name of a field located in the format buffer, or both.
- A *stored procedure* executes when the database management system (that is, Adabas) receives a special user call.

The same parameters are passed to the subprogram whether it is a trigger or a stored procedure.

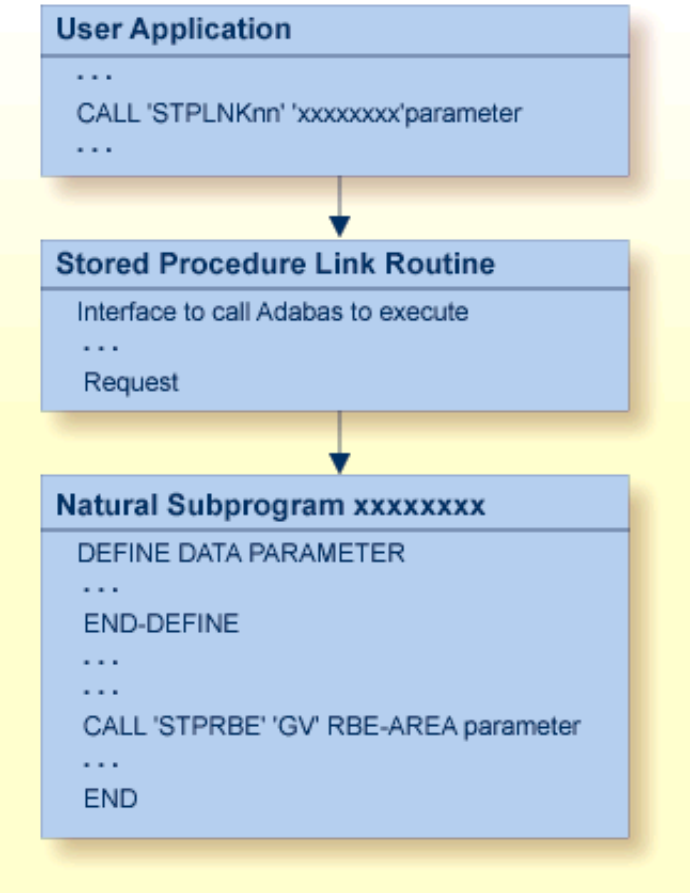
Stored Procedures

A stored procedure is invoked from a user application, which issues a special call to the procedure and may pass it one or more parameters. The procedure is executed by Adabas.

Because the procedure is stored in the Natural system file (an Adabas file) loaded on the database (server), it reduces the amount of data traffic to and from the server.

A sample stored procedure subprogram is provided; the call structure that is predefined in the stored procedure front-end may be modified.

The following figure illustrates stored procedure usage. STPLNK is the stored procedure link routine used to invoke a stored procedure request; STPRBE is the record buffer extraction routine, which is called by the procedure and used to retrieve the parameters passed by the calling routine.



Stored Procedure Usage

For more information about STPLNK, see the section *Stored Procedure Link Routine (STPLNKnn)*. For programming information about STPRBE, see the section *Record Buffer Extraction Routine (STPRBE)*.

Triggers

A trigger has two parts: the triggering event and the triggered procedure.

The triggering event is defined by a set of selection criteria such as an Adabas file number and optionally a command or a field name (or both). When the criteria are met, the event occurs and the triggered procedure is executed in response. For example, a trigger might be fired by an update command to the SALARY field in the EMPLOYEES file.

Note:
A trigger cannot be fired by another trigger.

Triggers can be defined to execute before or after the initiating Adabas command is executed by the Adabas nucleus. The behavior of a trigger depends to some extent on whether the trigger and the Adabas command are synchronized and, if so, whether the trigger participates in the transaction logic of the command.

Triggers have three major characteristics:

- pre-command or post-command execution.

  A trigger can execute before or after the initiating Adabas command.

- asynchronous or synchronous execution.

  A trigger can execute independently of Adabas command processing or require suspension of Adabas command processing (that is, command processing must wait for the results of the triggered procedure).

- participation or non-participation.

  A synchronous trigger may participate or not participate in the user’s transaction (that is, ET) logic.

Pre-command or Post-command

A *pre-command* (or "pre-") trigger is executed before the initiating Adabas command is processed by the Adabas nucleus. Before an Adabas command is executed, a check is made to verify whether a trigger should be fired.

For example, if a trigger is defined to fire every time an N1 command is issued against a specified file, the N1 is the initiating Adabas command. Before the N1 is executed, a check determines that a pre-command trigger is to be fired. The N1 is processed after the triggered procedure is successfully executed.

A *post-command* (or "post-") trigger executes after the initiating Adabas command is processed by the Adabas nucleus. The triggered procedure executes only if the return code for the command from the Adabas nucleus is zero. If the return code is nonzero, no checking for triggers is done, and processing continues in the normal way. For a successfully executed command, the command is checked for any triggers before processing of the command completes.

For example, if a trigger is defined to fire every time an L3 command is issued against a specified file, the L3 is the initiating Adabas command. After a zero return code is received, a check determines that a post-command trigger is to be fired. The triggered procedure is executed after the command is successfully executed and before the user is notified.

Asynchronous or Synchronous

An *asynchronous* trigger executes independently of the initiating Adabas command. Adabas command processing for the user continues uninterrupted while the triggered procedure executes as a separate process. The triggered procedure may execute after the user has already received the response from the initiating command.

When a command is issued that fulfills trigger criteria, the trigger is fired and processing of the Adabas command resumes. The Adabas command and the triggered procedure do not affect each other directly.

Asynchronous triggers are used when there is no interdependency between the procedure and the actual Adabas command.

A *synchronous* trigger requires a suspension of Adabas command processing for the user. The initiating Adabas command is suspended until the triggered procedure completes execution. It is possible that the results of the procedure will affect the Adabas command.

For a post-command trigger, the Adabas command executes before the triggered procedure and then is suspended; the results of the command are not returned to the user until the triggered procedure completes execution.

Participating or Non-participating

Synchronous triggers may participate or not in the logic of the initiating Adabas command.

A *participating* trigger results in the execution of a procedure that is assigned the same user (communication) ID as the Adabas command that caused the participating trigger to be fired; thus, it can participate fully in the logic of the transaction.

- An ET (end transaction) or BT (backout transaction) issued by the initiating command's process affects any transactions in flight from the trigger.
- An ET or BT issued by the triggered procedure affects any transactions in flight for the initiating command.

A *non-participating* trigger has a user (communication) ID that is different from that of the Adabas user queue element (UQE) that identifies the initiating command; thus, it does not participate in the initiating command's transaction logic.

- An ET or BT issued by the initiating command's process does not affect the triggered procedure.
- An ET or BT issued by the triggered procedure does not affect the initiating command's process.

Components

Adabas uses three major components to implement and process triggers and stored procedures:

- the Adabas trigger driver is part of the Adabas nucleus and has overall control of triggers and stored procedures. It detects procedure requests and initializes the Natural trigger driver to execute them.
- the Natural trigger driver runs as a batch Natural nucleus that actually executes both triggers and stored procedures. Operating in conjunction with the Adabas trigger driver, it handles the interprocess communications between the Adabas nucleus and the Natural subsystem that executes the procedure.
- Trigger Maintenance is a Natural application accessible from Adabas Online System (AOS). A system of menus allows you to create and maintain trigger definitions, define the triggers profile, and monitor and to some extent control trigger activity within the nucleus.

Adabas Trigger Driver

The Adabas trigger driver is executed as a part of the Adabas nucleus and generally controls the whole run-time processing of a trigger. It determines whether a trigger is to be fired, initiates the Natural trigger driver, and interacts with it to ensure the correct and timely processing of the procedures. For more detailed information about how procedures are processed, see section *Processing and Performance*.

Initialization

When the Adabas nucleus starts, it determines whether the ADARUN parameter SPT=YES has been specified; if so, the nucleus passes control to the Adabas trigger driver to allow it to initialize. During initialization, the Adabas trigger driver

- acquires storage for buffers;
- verifies that a valid trigger file is loaded;
- verifies the triggers profile on the trigger file and extracts the session parameters to be used in processing triggers and stored procedures for the session;
- verifies that the trigger file contains at least one trigger definition (a requirement for Adabas trigger driver initialization); and
- reads the trigger definitions and adds entries to the trigger table, which occupies a buffer in memory. In a nucleus cluster environment, the trigger table is obtained from an active nucleus. During procedure processing, the trigger table can then be checked for the existence of a trigger, instead of the more expensive alternative of reading the trigger file.

Starting Natural Subsystems

After the Adabas trigger driver is initialized, it starts the Natural subsystems that are responsible for the actual execution of the procedures.

The Natural subsystems execute user-written procedures. The maximum subsystems parameter in the Adabas triggers profile determines the number of Natural subsystems (1-10) to be started.

Each Natural subsystem is typically a minimally modified batch Natural nucleus that runs as a subtask in the Adabas address space. This affects the region size specified on the MPM startup JCL/JCS. The effect may be minimized by using a split Natural nucleus.

When a Natural subsystem is started, the Adabas trigger driver keeps track of any change in subsystem status or activity. The user can monitor these activities by using the Subsystem Activity function of the Triggers Maintenance facility.

When a Natural subsystem becomes active:

- the Natural trigger driver acquires control; and
- the Adabas trigger driver is informed that the subsystem is ready to start processing any procedures that may result from a stored procedure request or the firing of a trigger.

Checking for Triggers

For each command that the nucleus receives, the Adabas trigger driver determines whether a trigger needs to be fired.

For pre-command triggers, the Adabas trigger driver checks for triggers before the command is selected for processing by the Adabas thread. Once a command has been processed successfully by Adabas and the response code is zero, the Adabas trigger driver determines whether there are any post-command triggers to be fired. Only two triggers (one pre- and one post-command trigger) can be fired for any one command, regardless of the results.

If a command results in a trigger being fired, or if the Adabas trigger driver determines that the command is a stored procedure request, an entry is created in the

- pre-trigger queue if the command has not been executed; or the
- post-trigger queue if the command has been executed successfully.

The entry contains information obtained from both the command and the corresponding entry in the trigger table.

If a Natural subsystem is waiting for work, it is given the trigger request immediately. Otherwise, the trigger request remains in the pre- or post-trigger queue until the next subsystem is available. When a subsystem accepts a trigger request, processing continues under the control of the Natural trigger driver (see *Components*).

Processing Procedure Results

When the procedure completes execution, the Natural trigger driver places the results in the "trigger request entry" and the status is updated appropriately. When the Adabas trigger driver detects this, it takes responsibility for completing the trigger processing.

For both pre- and post-command triggers, the return code from the procedure determines how the results are processed. For more information, see the section *Processing the Results*.

Shutdown

The Adabas trigger driver keeps track of all Natural subsystems that fail. If all subsystems fail, it determines that no further processing of procedures is possible and terminates according to the error action value in the Adabas triggers profile. The error action Ignore or Reject in a nucleus cluster environment is passed to all other nuclei in the cluster.

The Adabas trigger driver also terminates if the Adabas nucleus receives the ADAEND or HALT operator command and instructs the Adabas trigger driver to shut down as well.

Natural Trigger Driver

The Natural trigger driver is initialized during the startup of the Natural subsystems. It is responsible for executing all triggered and stored procedures, and includes the following components:

STP	is invoked when a Natural subsystem is started. STP initializes the global data area STPGDA and establishes any necessary settings for the Natural session. Its primary function is to handle recovery from any errors and ensure that a restart is done.
STPPDRIV	functions as the main routine for the Natural trigger driver, and is responsible for invoking the procedure. STPPDRIV calls STPNAT.
STPNAT	is responsible for any communication with the Adabas trigger driver and for servicing the trigger requests for the subsystem. It notifies the Adabas trigger driver that it is ready for work.
SPAENA	(BS2000 only) retrieves the address of the database command queue.

Setting Up the Parameter List

When a trigger request is accepted by a Natural subsystem, STPPDRIV sets up the parameter list to be passed to the procedure, depending on which parameter list was specified by the trigger that was fired.

Invoking the Tracking Routine STPUTRAK

If the log trigger activity setting in the Adabas triggers profile is active, the routine STPUTRAK is invoked.

STPUTRAK is a user-defined routine that tracks every request to invoke a procedure, both before and after the procedure is invoked. You can use this routine to write trace messages or audit trigger processing for each subsystem. A default STPUTRAK routine is supplied.

Parameters that are similar to those of the procedure and contain information about the trigger are passed to the STPUTRAK routine, in addition to a 250-byte area that can be used as a work area. STPUTRAK uses the work area to retain information for the entire session; the work area is never changed by the Natural trigger driver. STPUTRAK contains an option that allows you to pass this work area to the procedure.

Processing the Procedure

After STPUTRAK has been processed and control has returned to STPPDRIV, the triggered procedure is invoked with a CALLNAT.

When the procedure completes processing, STPPDRIV again checks whether the "log trigger activity" option in the Adabas triggers profile is set to active before informing STPNAT of the results. If log activity is active, STPUTRAK is invoked so that the results of the procedure can be audited.

Recovering from Errors

The procedure does not terminate normally if it incurs an error such as NAT0954, NAT3009, or NAT1305. Instead, the Natural trigger driver performs error recovery.

Regardless of the log trigger activity setting, information about the error is conveyed to STPUTRAK. This information can be used by the database administrator (DBA) for debugging or problem analysis. Therefore, STPUTRAK should always be available for execution by the Natural trigger driver. Software AG recommends activating the trace to obtain this information.

Updating the Trigger Request Entry

After the results of a procedure are delivered to STPNAT, the trigger request is updated and its status is changed to "completed". When the Adabas trigger driver detects the updated entry, it takes responsibility for completing the trigger processing.

STPNAT waits for another trigger request to be made, and the entire cycle starts again.

Trigger Maintenance

Trigger Maintenance, which requires the full version of Adabas Online System, is an interactive facility for creating trigger definitions and monitoring system processing. For detailed information, see the section *Trigger Maintenance* in this documentation.

A *trigger definition* comprises the name and attributes of the procedure to be executed, and the selection criteria that compose the *triggering event* (Adabas command type, file name or number, and field name).

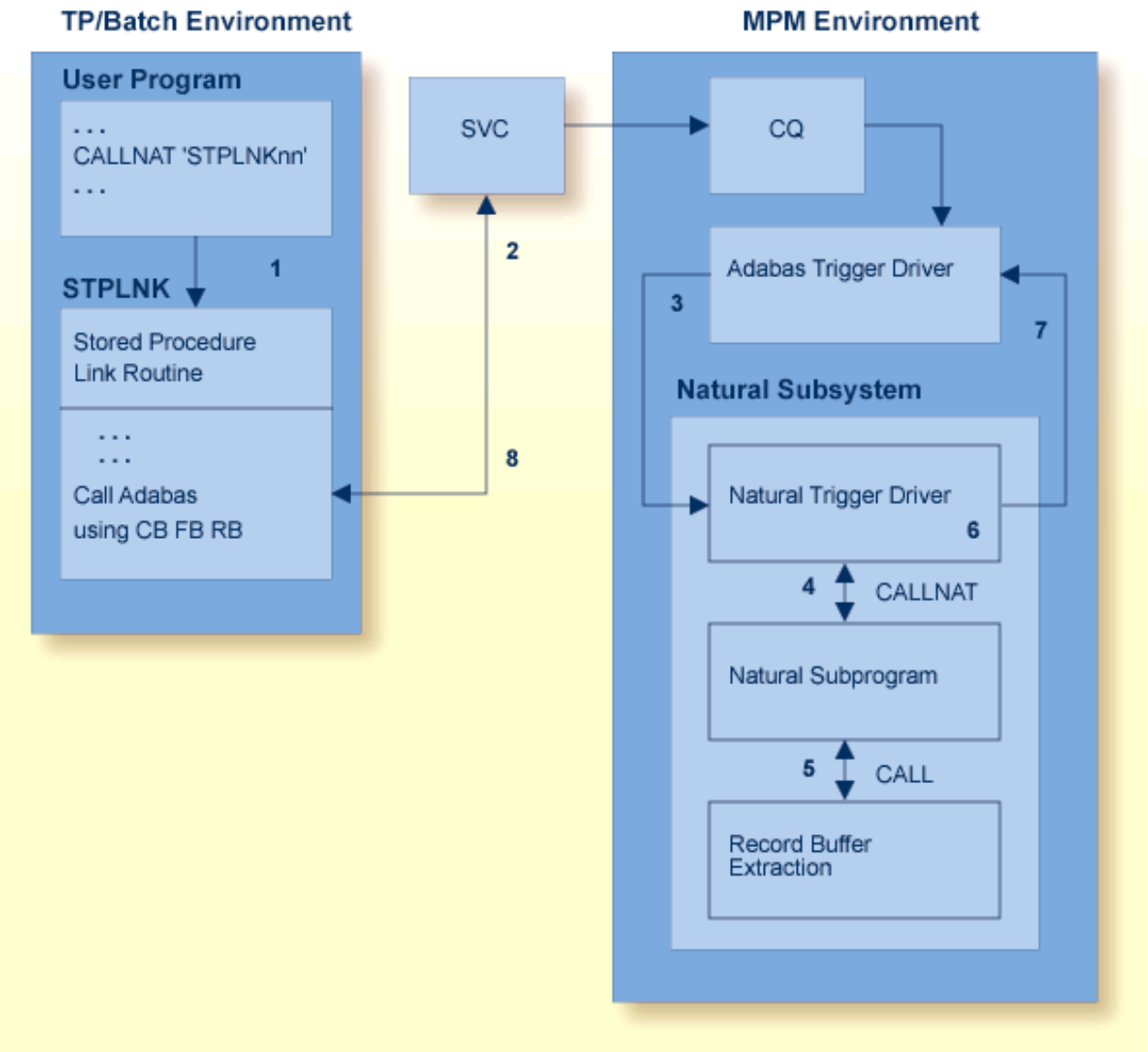
Trigger maintenance also provides functions for monitoring and controlling the execution of any trigger processing by the nucleus. You can examine the activities of both the Adabas and Natural trigger drivers and

- determine the state of the triggers environment in the nucleus; that is, determine which triggers are active and which are not, buffer sizes, and the status of the Natural subsystems;
- modify the settings of various parameters such as activity timeout, log trigger activity, and error action during an active session;
- examine the pre- and post-trigger queues, which contain the procedures waiting to be executed for the pre- and post-command triggers that have been fired;
- examine the activity of the Adabas triggers and stored procedures facility to determine what is executing, whether problems exist, the number of Natural subsystems that are currently active, and so on;
- refresh the trigger table in the Adabas nucleus with new, deleted, or updated trigger definitions; and
- individually activate or deactivate a trigger. If a problem is found with a particular trigger, it can be temporarily or permanently deactivated while the problem is resolved.

Processing Summary

Stored Procedure Processing

The steps involved in stored procedure processing are illustrated in the following figure and described in *Processing Steps*.



Stored Procedure Processing

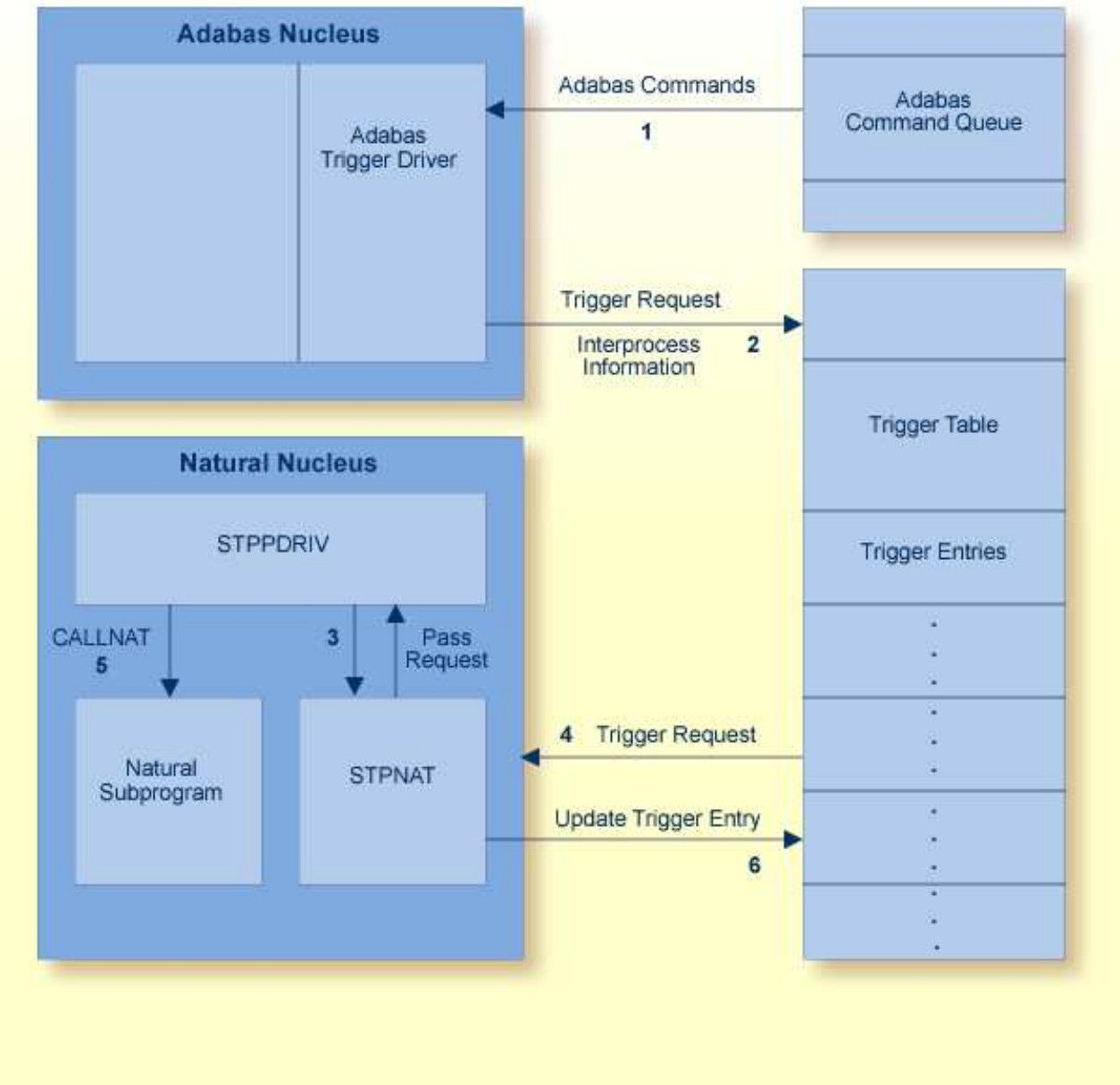
Processing Steps

1. The user application sets up the required parameters and issues a CALL to the stored procedure link routine STPLNK*nn*.
2. STPLNK*nn* interprets the user's request and issues the call to the DBMS in the form of a standard Adabas API (direct call).
3. The Adabas trigger driver receives the stored procedure request and passes it on to the Natural trigger driver.
4. The specified Natural subprogram (the stored procedure) is invoked.
5. The caller's parameters are made available to the subprogram by the record buffer extraction routine, as required. The parameters can be modified if the option settings permit it.

6. On completion, the Natural subprogram returns control to the Natural trigger driver.
7. The Natural trigger driver returns the results of the stored procedure request to the Adabas trigger driver.
8. The user is notified and any modified parameters are returned.

Trigger Processing

The steps involved in trigger processing are illustrated by the following figure and described in *Processing Steps*.



Trigger Processing

Processing Steps

1. The Adabas trigger driver receives all Adabas commands and inspects them for trigger events.
2. If a command meets trigger criteria, the Adabas trigger driver places the trigger request and interprocess information in the trigger table.
3. When the Natural trigger driver control routine STPPDRIV is ready to process another trigger request, it calls STPNAT.
4. STPNAT gets the next trigger request from the trigger table and passes it to STPPDRIV.
5. STPPDRIV determines the trigger request type and issues a CALLNAT to the Natural subprogram named in the trigger definition, passing the relevant parameters.
6. STPNAT updates the trigger entry in the trigger table to indicate when the trigger is completed.