Creating triggers

http://manuals.sybase.com/onlinebooks/group-as/asg1250e/sqlug/@Generic__BookTextView/47981;pt=47865

A trigger is a database object. When you create a trigger, you specify the table and the data modification commands that should "fire" or activate the trigger. Then, you specify the action(s) the trigger is to take.

For example, this trigger prints a message every time anyone tries to insert, delete, or update data in the *titles* table:

```
create trigger t1
on titles
for insert, update, delete
as
print "Now modify the titleauthor table the same way."
```

Except for the trigger named *deltitle*, the triggers discussed in this chapter are not included in the *pubs2* database shipped with Adaptive Server. To work with the examples shown in this chapter, create each trigger example by typing in the **create trigger** statement. Each new trigger for the same operation--**insert**, **update** or **delete**--on a table or column overwrites the previous one without warning, and old triggers are dropped automatically.

create trigger syntax

Here is the complete **create trigger** syntax:

```
create trigger [owner.]trigger_name
     on [owner.]table_name
     {for {insert, update, delete}
     as SQL_statements
```

Or, using the **if update** clause:

```
create trigger [owner.]trigger_name
     on [owner.]table_name
     for {insert, update}
     as
          [if update (column_name )
               [{and | or} update (column_name)]...]
               SQL_statements
          [if update (column_name)
               [{and | or} update (column_name )]...
               SQL_statement ]...
```

The **create** clause creates and names the trigger. A trigger's name must conform to the rules for identifiers.

The **on** clause gives the name of the table that activates the trigger. This table is sometimes called the trigger table.

A trigger is created in the current database, although it can reference objects in other databases. The owner name that qualifies the trigger name must be the same as the one in the table. Only a table owner can create a trigger on a table. If the table owner is given with the table name in the **create trigger** clause or the **on** clause, it must also be specified in the other clause.

The **for** clause specifies which data modification commands on the trigger table activate the trigger. In the earlier example, an **insert**, **update**, or **delete** to *titles* makes the message print.

The SQL statements specify trigger conditions and trigger actions. Trigger conditions specify additional criteria that determine whether **insert**, **delete**, or **update** causes the trigger actions to be carried out. You can group multiple trigger actions in an **if** clause with **begin** and **end**.

An **if update** clause tests for an insert or update to a specified column. For updates, the **if update** clause evaluates to true when the column name is included in the **set** clause of an **update** statement, even if the update does not change the value of the column. Do not use the **if update** clause with **delete**. You can specify more than one column, and you can use more than one **if update** clause in a **create trigger** statement. Since you specify the table name in the **on** clause, do not use the table name in front of the column name with **if update**.

SQL statements that are not allowed in triggers

Since triggers execute as part of a transaction, the following statements are not allowed in a trigger:

- All **create** commands, including **create database**, **create table**, **create index**, **create procedure**, **create default**, **create rule**, **create trigger**, and **create view**
- All **drop** commands
- **alter table** and **alter database**
- **truncate table**
- **grant** and **revoke**
- **update statistics**
- **reconfigure**
- **load database** and **load transaction**
- **disk init**, **disk mirror**, **disk refit**, **disk reinit**, **disk remirror**, **disk unmirror**
- **select into**