THE DYNAMIC PROCESSING OF THE DATABASES USING TRIGGERS

ION LUNGU, ADRIAN GHENCEA
THE BUCHAREST ACADEMY OF ECONOMIC STUDIES
6 PIAȚA ROMANĂ, BUCHAREST 010374
"TITU MAIORESCU" UNIVERSITY, BUCURESTI
22 DÂMBOVNICULUI STREET, BUCHAREST, ROMANIA. 040051
ion.lungu@ie.ase.ro, adighencea@gmail.com

Abstract:
Triggers are used for dynamic processing of the relational databases systems. It is based upon the notion of event-condition-action, respectively on rules that related to the three components aforementioned. Those are 'active rules' initially defined in the active databases domain. An active database system offers the designer of an application the possibility of monitoring the occurrence of specific types of events that occur in the database. Those events are usually database changes, but they can also be mentions (invocations) of procedures and functions, or even events occurrence generated by the system clock. When such event occurs, an active rule prompts the assessment of a condition. The rule action is executed if the result is true.

Key words: *active database, triggers, stored procedures*

JEL classification: *C00, C43, C61, C63, M15*

When are triggers used?

According to Garth Wells[1], "a trigger is a database object which is attached to a table. In many aspects, it is similar to a stored procedure". More accurately, triggers most often refer to "a special type of stored procedures". The main difference between a trigger and a stored procedure is that the trigger is attached to a table and is only activated when an INSERT, UPDATE or DELETE action takes place. We can therefore state that the modifying action takes place only when the trigger has been created and activated.
The next short example shows us how to create a trigger which displays the system lock when a row is inserted in the table to which it is attached.

```
CREATE TABLE Source (Sou_ID int IDENTITY, Sou_Desc varchar(10))
go
CREATE TRIGGER tr_Source_INSERT
ON Source
FOR INSERT
AS
PRINT GETDATE()
go
INSERT Source (Sou_Desc) VALUES ('Test 1')
```

Because the trigger is in the database and anyone with the necessary rights can use it, it allows us to write a SQL instruction set which multiple applications can use. Therefore, this allows the avoidance of redundant code when multiple applications need to perform operations within the same database.
We can use triggers to perform different actions, such as:

- Creating a database audit list. For example, the updates of the database table *Commands* through the corroboration of the information from the audit list;

- Applying a business rule. For example, it can be established when a command exceeds the credit limit allowed for a client, displaying a message in this respect;
- Obtaining additional data which is not available in a table or in a database. For example, when there is an undergoing update on a table column named *quantity*, you can calculate the column *total_price;*
- Enforcement of referential integrity. When a client is deleted, you can use a trigger to delete the corresponding rows (the rows with the same client code), within the *Commands* table.

The aforementioned examples are known as DML Triggers (Data Manipulation Language), since triggers are defined as part of the data manipulation language and are as such executed when the data is being processed. Some systems also support non-data triggers, through events such as table creation, authentication etc. Therefore, DML triggers can be used on audit purposes.
The major characteristics of the database triggers and their effects are as follows:

- Triggers do not allow parameters and arguments (but can stock affected date in temporary tables);
- Triggers cannot perform rollback actions since they are a part of the SQL Trigger declaration (only through autonomous transactions);
- Triggers can cancel an on-going operation;
- Triggers can provoke errors in the table.

Triggers offer a high degree of flexibility to the database administrators and developers. They are simply stored procedures which can be configured to run automatically, when certain events take place. At all levels, relational database use triggers in one form or another.
We will analyse the syntaxes which Microsoft SQL Server uses, as well as the Oracle ones. However, the basic concepts are applied equally in both variants. The differences are more keyword and formatting related.
SQL Server triggers versus Oracle triggers
At the level of the **SQL Server**, triggers can be defined[2] in two different ways:

- External triggers;
- SQL triggers.

For external triggers, the command CRTPFTRG CL is used. The program contains a set of triggers which can be defined in any language. The external triggers are: Insert, Update, Delete or Read. For SQL triggers we use CREATE TRIGGER. The triggering program is defined fully using SQL. The SQL triggers are: Insert, Update, and Delete.
Once a trigger is associated to a table, it can be used every time a change operation is initiated on the table. The SQL and external triggers can be defined for the same table, up to 200 triggers for a single table.
The DML triggers are executed when a user is trying to modify the data through an event in the data manipulation language.
Starting with the 2005 MS SQL Server version, support for DDL (Data Definition Language) triggers has been implemented. The DDL triggers, same as normal triggers, launch stored procedures as a response to an event. The DDL and DML triggers are used for different purposes. The DML triggers operate with Insert, Update and Delete and contribute to the rules compliance and the extension of the data integrity when they are modified in a table.

The DDL triggers operate with Create, Alter, Drop and other stored procedures which use similar operations. Those are used in order to perform administrative tasks and apply rules which affect databases. Those apply to certain types of commands for a server or a database.
The DML and DDL triggers can *create* and *modify* through the usage of Transact-SQL[3]. Therefore, in SQL Server triggers are grouped in multiple categories:

- Multiple triggers – in this case SQL Server allows the creation of triggers for each DML, DDL or LOGON event. For example, if the trigger creates the Update action, a table with a similar trigger is made, thus a similar Update trigger is also created. In previous versions of SQL Server, there only was a single trigger for each Insert, Update, and Delete action;
- Recursive triggers – SQL Server allows the recursive invocation of triggers when the Recursive_Trigger setting is activated using Alter Database;
- Nested triggers – the triggers can be nested to a maximum of 32 levels. If a trigger causes a change in a table, a second trigger is created, which can then trigger a third, and so forth. If any trigger triggers an infinite loop, the nesting level is exceeded and the trigger is annulled.

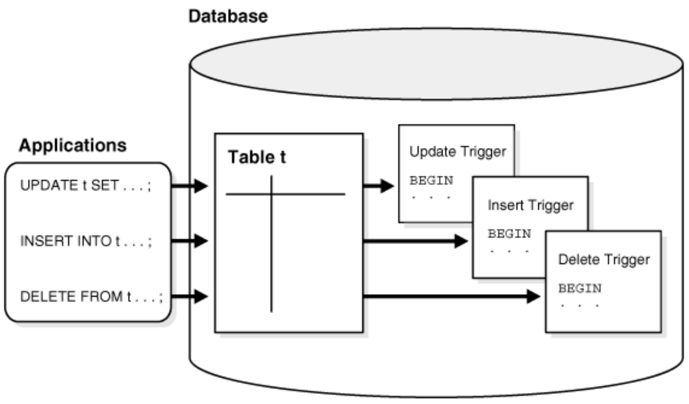

Fig 1. SQL Server triggers

SQL Server trigger limitations

Any application existent at this moment is limited to a certain degree. Therefore, SQL Server is no exception, having a number of conditions and limits from which we will mention the following:

- Create Trigger must always be the first statement, only being applied to a single table;
- A trigger is created only for the current database; however, a trigger may have as reference an object outside the current database;
- The name of the table and the trigger must be identical;
- The same action of a trigger can be defined by more than one user (for example, Insert and Update) in the same Create Trigger statement.

Any SET statement can be specified within a trigger. The activated SET option remains in operation during the trigger execute phase and after that returns to the initial setting. When a trigger is activated, the results are return at the application's request, same as with stored procedures. To 'prevent' the return of results as a result of activating a trigger, the SELECT statement, which returns a series of results, must not be used.
Using triggers, SQL Server has improved its concurrent access to data. From this perspective, triggers represent SQL Server characteristics based on the RLV technology which no longer require improvement.

In comparison to SQL Server, along with the fact that triggers activate on data modifying, **Oracle 9i** states that a trigger is activated when tables are modified and when events are created, both in online and offline mode. Those triggers are named 'The trigger level scheme' and contain: *after creation, before alter, after alter, before drop, after drop, before logoff, after logon.*
Triggers are, in this case, represented by a PL/SQL program, associated with a database table. The code from the trigger defines the database action for its data manipulation (using Insert, Update, and Delete).
Unlike stored procedures and functions, the database triggers are executed or called whenever the table is affected by any of the DML operations aforementioned. Up until Oracle 7.0 only 12 triggers could be associated to a certain table, but in more recent versions this limitation has been removed. A database executes triggers with privileges on the owner-side and not the client.
An Oracle database triggers has the following three components:

- A triggering event;
- A trigger constraint (optional);
- A trigger action.

Oracle allows an automatic trigger creation. This option can be either activated or deactivated using Alter_Tabel or Alter_Trigger. Triggers can be used in different ways in order to personalize different data and information on an Oracle database. For example, Oracle triggers are frequently used for:

- Allowing advanced auditing;
- Preventing invalid transactions;
- Entailing referential integrity;
- Allowing the application of complex business rules;
- Implementing complex business permits;
- Providing translucent logging events;
- Allowing the building of complex views that can be updated;
- Allowing events monitoring in the database.

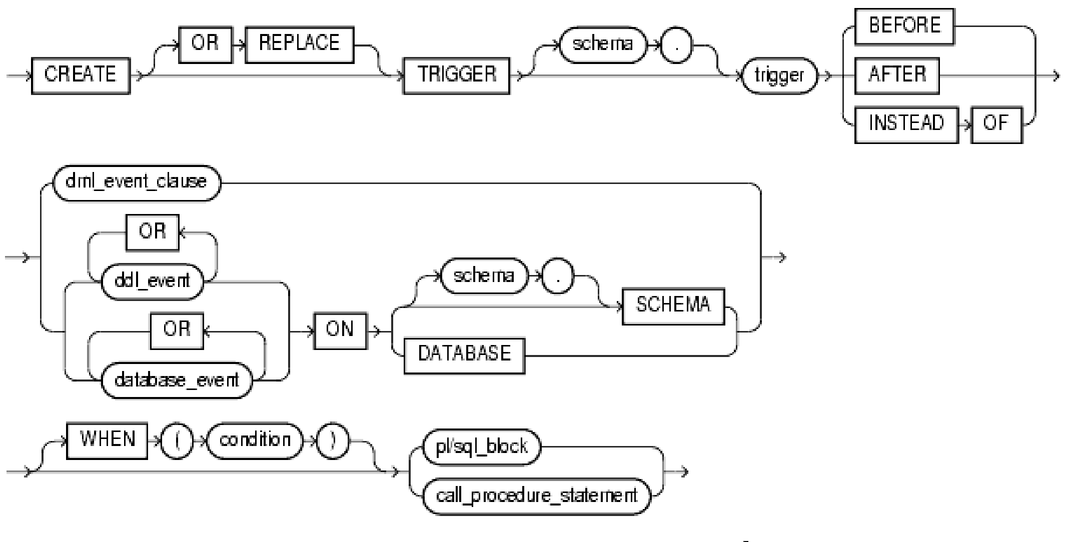

Fig 2. Creating a Trigger in Oracle
If you are asking yourself whether Oracle supports a SELECT trigger similar to the INSERT, UPDATE and DELETE triggers, then the answer is negative. However, you can achieve that by creating a BEFORE trigger in two steps. Firstly, a Handler (a

PL/SQL procedure) must be created, which will process Select events on which we desire to use the trigger, and after that we can define an 'audit policy' of the PL/SQL procedure previously created. For example:

```
PROCEDURE NAME_OF_SELECT_EVENT_HANDLER
( object_schema VARCHAR2
, object_name VARCHAR2
, policy_name VARCHAR2
);
```

The Handler can be used for SELECT events for multiple tables. When enabled, it will be informed in regards to the specified object – the scheme and the name – as well as the specified policy which was defined for said event. Any action can be realised using the Handler, including the definition of an autonomous transaction which saves an "audit-record"[4].
Making a comparison between SQL Server and Oracle Database 9i is no easy task. The performance of the databases depends more on the experience of the developers and its administrators than on the database provider. Both of them offer ANSI SQL-92 'entry level' and 'intermediate level' support. The SQL dialect supports SQL Server and is named Trasact-SQL. The SQL dialect supported by Oracle Database 9i is named PL/SQL, which is a more powerful language than T-SQL.

Description	**PL/SQL**	**T-SQL**
Indexes	B-Tree indexes, Bitmap indexes, Partitioned indexes, Function-based indexes, Domain indexes	B-Tree indexes
Tables	Relational tables, Object tables, Temporary tables, Partitioned tables, External tables, Index organized tables	Relational tables, Temporary tables
Triggers	BEFORE triggers, AFTER triggers, INSTEAD OF triggers, Database Event triggers	AFTER triggers, INSTEAD OF triggers
Procedures	PL/SQL statements, Java methods, third-generation language (3GL) routines	T-SQL statements
Arrays	PL/SQL statements, Java methods, third-generation language (3GL) routines	Not Supported

Table 1 – Comparison between PL/SQL and T-SQL

Conclusions

Both products can be used in a stable and efficient system construction. The stability and the efficiency of the applications and the databases depend on the experience of the developers rather than the database provider. However, SQL Server does have some advantages in comparison to Oracle 9i, and vice-versa.

SQL Server advantages:

- SQL Server is less expensive than Oracle Database 9i;
- SQL Server presents higher a price/performance ratio;
- SQL is generally accepted as easier to install, utilize and administrate.

Oracle 9i Database advantages:

- Oracle 9i Database supports all known platforms, not only the Windows-based ones;
- PL/SQL is a far more powerful language than T-SQL;
- It allows more parameters for settings.

When triggers are used correctly, they can save a large number of developing activities. We can therefore conclude that one of the main advantages in each of the presented cases is that they are stored within the database, which means that they can be used in all client/webpages applications at which a database can connect.

REFERENCES

1. Arup Nanda, *Fine-Grained Auditing for Real-World Problems, Part 3*, series of three articles in Oracle Magazine, 2010
2. Garth Wells, *Code Centric: T-SQL Programming with Stored Procedures and Triggers – Apress, 2001, ISBN-13: 978-1893115835*
3. IBM eServer iSeries Information Center, Version 5 Release (V5R3), http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp
4. Lucas Jellema, *Select Trigger in Oracle Database – introducing Fine Grained Auditing*, AMIS – Weblog for the AMIS Techology corner, 2010
5. MSDN Library, http://msdn.microsoft.com/en-us/library/ms189599.aspx
6. Oracle 10gR2 Documentation Library PL/SQL Packages and Types Reference - DBMS_FGA
7. Oracle® Database Security Guide 10g Release 2 (10.2) - 12 Configuring and Administering Auditing
8. SQL Server 2000 Books Online