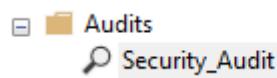


Auditorias

En SQL Server.

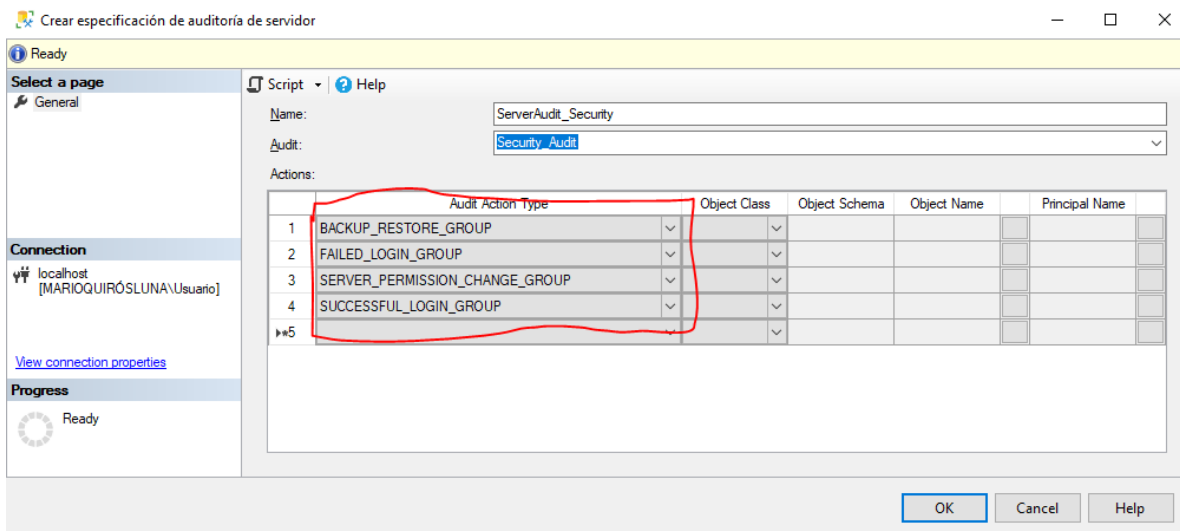
Se crea una auditoria.



Se activa.

```
ALTER SERVER AUDIT [Security_Audit] WITH (STATE = ON)
GO
```

Se agregan las acciones del servidor a monitorear para esa auditoria.



Se activan las especificaciones.

Ahora para comprobar si funciona.

Se realiza un backup.

```
BACKUP DATABASE [Testing]
TO DISK='C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\Backup\TestAudit.bak'
```

El backUp se realizó con éxito.

Messages

Processed 584 pages for database 'Testing', file 'Testing' on file 1.
Processed 1 pages for database 'Testing', file 'Testing_log' on file 1.
BACKUP DATABASE successfully processed 585 pages in 0.191 seconds (23.895 MB/sec).

Completion time: 2022-06-14T11:08:37.5970129-06:00

Se visualiza el Log de la auditoria, para comprobar que salgan las acciones realizadas, en este caso el backUp realizado.

Log file summary: No filter applied						
Date	Event Time	Server Instance Name	Action ID	Class Type	Sequence Number	Succeeded
✓ 14/6/2022 17:09:09	17:09:09.6582440	MARIOQUIRÓSLUNA	LOGIN SUCCEEDED	LOGIN	1	True
✓ 14/6/2022 17:08:37	17:08:37.1918992	MARIOQUIRÓSLUNA	BACKUP	DATABASE	1	True
✓ 14/6/2022 17:08:04	17:08:04.0176741	MARIOQUIRÓSLUNA	LOGIN SUCCEEDED	LOGIN	1	True
✓ 14/6/2022 17:08:03	17:08:03.8449153	MARIOQUIRÓSLUNA	LOGIN SUCCEEDED	LOGIN	1	True
✓ 14/6/2022 17:05:29	17:05:29.1184520	MARIOQUIRÓSLUNA	LOGIN SUCCEEDED	LOGIN	1	True
✓ 14/6/2022 17:05:28	17:05:28.6761678	MARIOQUIRÓSLUNA	LOGIN SUCCEEDED	LOGIN	1	True
✓ 14/6/2022 17:05:11	17:05:11.1487855	MARIOQUIRÓSLUNA	LOGIN SUCCEEDED	LOGIN	1	True
✓ 14/6/2022 17:05:11	17:05:11.0318309	MARIOQUIRÓSLUNA	LOGIN SUCCEEDED	LOGIN	1	True
✓ 14/6/2022 17:03:21	17:03:21.2249699	MARIOQUIRÓSLUNA	LOGIN SUCCEEDED	LOGIN	1	True
✓ 14/6/2022 17:03:01	17:03:01.6186138	MARIOQUIRÓSLUNA	LOGIN SUCCEEDED	LOGIN	1	True
✓ 14/6/2022 16:53:14	16:53:14.7491264	MARIOQUIRÓSLUNA	AUDIT SESSION CHANGED	SERVER AUDIT	1	True

Se crea otra auditoria, pero esta para la base de datos en específico que deseamos monitorear.

Audit name: Database_Audit

Queue delay (in milliseconds): 1000

On Audit Log Failure:

- ☒ Continue
- ☐ Fail operation
- ☐ Shut down server

Audit destination: File

Path: C:\Program Files\Microsoft SQL Server\MSSQL15.M ...

Audit File Maximum Limit:

- ☒ Maximum rollover files:
 - ☒ Unlimited
 - ☐ Maximum files:
Number of files: 2147483647
- ☐ Maximum file size:
Maximum file size: 0 MB GB TB
 - ☒ Unlimited

☐ Reserve disk space

Se agregan las acciones de la base de datos especifica a monitorear para esa auditoria.

Name: DatabaseAudit_Actions


Audit: Database_Audit

Actions:


	Audit Action Type	Object Class	Object Schema	Object Name	Principal Name
▶ 1	DELETE	DATABASE		Testing	public
2	INSERT	DATABASE		Testing	public
3	SELECT	DATABASE		Testing	public
4	UPDATE	DATABASE		Testing	public
* 5					

Se activa las especificaciones.

Enable Database Audit Specification

 **Success** 1 Total
1 Success

Details:

Action	Status	Messa
 Enable Database Audit Specification 'D...	Success	

Close

Para comprobar si funciona la auditoria, se realizan las acciones monitoreadas.

```

USE [Testing]
INSERT INTO [MOCKS].[MOCK_DATA_FIRST_NAMES]
(id, first_name)
VALUES
(501, 'JACINTO')

INSERT INTO [MOCKS].[MOCK_DATA_LAST_NAMES]
(id, last_name)
VALUES
(501, 'BASURILLA')

UPDATE [MOCKS].[MOCK_DATA_FIRST_NAMES]
SET [first_name] = 'Jacinto'
WHERE id = 501

SELECT * FROM [MOCKS].[MOCK_DATA_FIRST_NAMES] ORDER BY id DESC
SELECT * FROM [MOCKS].[MOCK_DATA_LAST_NAMES] ORDER BY id DESC

DELETE FROM [MOCKS].[MOCK_DATA_FIRST_NAMES]
WHERE id = 501

DELETE FROM [MOCKS].[MOCK_DATA_LAST_NAMES]
WHERE id = 501

```

Verificamos el Log de la auditoria.

Log file summary: No filter applied						
Date	Action ID	Class Type	Databa...	Object Name	Statement	
14/6/2022 18:09:58	SELECT	TABLE	Testing	MOCK_DATA_LAST_NAMES	DELETE FROM [MOCKS].[MOCK_DATA_LAST_NAMES]	WH
14/6/2022 18:09:58	DELETE	TABLE	Testing	MOCK_DATA_LAST_NAMES	DELETE FROM [MOCKS].[MOCK_DATA_LAST_NAMES]	WH
14/6/2022 18:09:58	SELECT	TABLE	Testing	MOCK_DATA_FIRST_NAMES	DELETE FROM [MOCKS].[MOCK_DATA_FIRST_NAMES]	W
14/6/2022 18:09:58	DELETE	TABLE	Testing	MOCK_DATA_FIRST_NAMES	DELETE FROM [MOCKS].[MOCK_DATA_FIRST_NAMES]	W
14/6/2022 18:09:50	SELECT	TABLE	Testing	MOCK_DATA_LAST_NAMES	SELECT * FROM [MOCKS].[MOCK_DATA_LAST_NAMES] ORDE	
14/6/2022 18:09:50	SELECT	TABLE	Testing	MOCK_DATA_FIRST_NAMES	SELECT * FROM [MOCKS].[MOCK_DATA_FIRST_NAMES] ORDE	
14/6/2022 18:09:45	UPDATE	TABLE	Testing	MOCK_DATA_FIRST_NAMES	UPDATE [MOCKS].[MOCK_DATA_FIRST_NAMES] SET first_r	
14/6/2022 18:09:45	SELECT	TABLE	Testing	MOCK_DATA_FIRST_NAMES	UPDATE [MOCKS].[MOCK_DATA_FIRST_NAMES] SET first_r	
14/6/2022 18:08:49	SELECT	VIEW	Testing	tables	SELECT SCHEMA_NAME(tbl.schema_id) AS [Schema], tbl.name /	
14/6/2022 18:08:09	INSERT	TABLE	Testing	MOCK_DATA_LAST_NAMES	INSERT INTO [MOCKS].[MOCK_DATA_LAST_NAMES] (id, last_n	
14/6/2022 18:08:09	INSERT	TABLE	Testing	MOCK_DATA_FIRST_NAMES	INSERT INTO [MOCKS].[MOCK_DATA_FIRST_NAMES] (id, first	
14/6/2022 18:05:42	SELECT	TABLE	Testing	MOCK_DATA_LAST_NAMES	SELECT * FROM [MOCKS].[MOCK_DATA_LAST_NAMES] ORDE	
14/6/2022 18:05:42	SELECT	TABLE	Testing	MOCK_DATA_FIRST_NAMES	SELECT * FROM [MOCKS].[MOCK_DATA_FIRST_NAMES] ORDE	
14/6/2022 18:05:16	SELECT	TABLE	Testing	MOCK_DATA_FIRST_NAMES	SELECT TOP 1 * FROM [MOCKS].[MOCK_DATA_FIRST_NAMES	

Se puede ver que se realizo la consulta, si realizamos scroll a la derecha, se mostraran columnas donde dirá a que tabla, que esquema, que usuario y demás cosas a monitorear sobre la consulta.

En PostgreSQL

En el caso de PostgreSQL se realiza de una manera similar a como lo hicimos en practica de clase, se debe crear una tabla donde se almacenen los datos que se quiere almacenar para la auditoria.

En este caso me base en el ejemplo que se muestra en un Blog, en el que realiza la auditoria por medio de triggers.

Primero se crea la tabla para la prueba, en donde se almacenarán los datos de la auditoria.

```
CREATE schema audit;

REVOKE CREATE ON schema audit FROM public;

CREATE TABLE audit.logged_actions (
    schema_name text NOT NULL,
    TABLE_NAME text NOT NULL,
    user_name text,
    action_tstamp TIMESTAMP WITH TIME zone NOT NULL DEFAULT CURRENT_TIMESTAMP,
    action TEXT NOT NULL CHECK (action IN ('I','D','U')),
    original_data text,
    new_data text,
    query text
) WITH (fillfactor=100);

REVOKE ALL ON audit.logged_actions FROM public;
```

Se dan los permisos a la tabla.

```
GRANT SELECT ON audit.logged_actions TO public;

CREATE INDEX logged_actions_schema_table_idx
ON audit.logged_actions(((schema_name||'.'||TABLE_NAME)::TEXT));

CREATE INDEX logged_actions_action_tstamp_idx
ON audit.logged_actions(action_tstamp);

CREATE INDEX logged_actions_action_idx
ON audit.logged_actions(action);
```

Se crea la función que realizara los insert cuando se realice alguna acción sobre la tabla que estamos auditando.

```

CREATE OR REPLACE FUNCTION audit.if_modified_func() RETURNS TRIGGER AS $body$
DECLARE
    v_old_data TEXT; v_new_data TEXT;
BEGIN
    IF (TG_OP = 'UPDATE') THEN
        v_old_data := ROW(OLD.*);
        v_new_data := ROW(NEW.*);
        INSERT INTO audit.logged_actions (schema_name,table_name,user_name,action,original_data,new_data,query)
        VALUES (TG_TABLE_SCHEMA::TEXT,TG_TABLE_NAME::TEXT,session_user::TEXT,substring(TG_OP,1,1),v_old_data,v_new_data, current_query());
        RETURN NEW;
    ELSEIF (TG_OP = 'DELETE') THEN
        v_old_data := ROW(OLD.*);
        INSERT INTO audit.logged_actions (schema_name,table_name,user_name,action,original_data,query)
        VALUES (TG_TABLE_SCHEMA::TEXT,TG_TABLE_NAME::TEXT,session_user::TEXT,substring(TG_OP,1,1),v_old_data, current_query());
        RETURN OLD;
    ELSEIF (TG_OP = 'INSERT') THEN
        v_new_data := ROW(NEW.*);
        INSERT INTO audit.logged_actions (schema_name,table_name,user_name,action,new_data,query)
        VALUES (TG_TABLE_SCHEMA::TEXT,TG_TABLE_NAME::TEXT,session_user::TEXT,substring(TG_OP,1,1),v_new_data, current_query());
        RETURN NEW;
    ELSE
        RAISE WARNING '[AUDIT.IF_MODIFIED_FUNC] - Other action occurred: %, at %',TG_OP,now();
        RETURN NULL;
    END IF;
EXCEPTION
    WHEN data_exception THEN
        RAISE WARNING '[AUDIT.IF_MODIFIED_FUNC] - UDF ERROR [DATA EXCEPTION] - SQLSTATE: %, SQLERRM: %',SQLSTATE,SQLERRM;
        RETURN NULL;
    WHEN unique_violation THEN
        RAISE WARNING '[AUDIT.IF_MODIFIED_FUNC] - UDF ERROR [UNIQUE] - SQLSTATE: %, SQLERRM: %',SQLSTATE,SQLERRM;
        RETURN NULL;
    WHEN OTHERS THEN
        RAISE WARNING '[AUDIT.IF_MODIFIED_FUNC] - UDF ERROR [OTHER] - SQLSTATE: %, SQLERRM: %',SQLSTATE,SQLERRM;
        RETURN NULL;
END;
$body$
LANGUAGE plpgsql
SECURITY DEFINER
SET search_path = pg_catalog, audit;

```

Se crea una tabla para realizar pruebas. Y a esta se le agrega un trigger con la función de la auditoria.

```

CREATE TABLE categories (
    category_id smallint NOT NULL,
    category_name character varying(15) NOT NULL,
    description text,
    picture bytea
);

```

```

CREATE TRIGGER t_if_modified_trg
AFTER INSERT OR UPDATE OR DELETE ON categories
FOR EACH ROW EXECUTE PROCEDURE audit.if_modified_func();

```

Se realizan unas pruebas.

```

INSERT INTO categories VALUES (1, 'Beverages', 'Soft drinks, coffees, teas, beers, and ales', '\x');
INSERT INTO categories VALUES (2, 'Condiments', 'Sweet and savory sauces, relishes, spreads, and seasonings', '\x');
DELETE FROM categories WHERE category_id = 1;
UPDATE categories SET description = '' WHERE category_id = 2;
UPDATE categories SET description = 'condiments 2' WHERE category_id = 2;

```

El resultado de la auditoria sería el siguiente.

	schema_name	table_name	user_name	action_timestamp	action	original_data	new_data	query
	text	text	text	timestamp with time zone	text	text	text	text
1	public	categories	postgres	2022-06-16 08:5...	I	[null]	(1,Beverages,'Soft drinks, coffees, teas, beers, and ales','')	INSERT INTO categories VALUES (1, 'Beverages',
2	public	categories	postgres	2022-06-16 08:5...	I	[null]	(2,Condiments,'Sweet and savory sauces, relishes, spreads, an...	INSERT INTO categories VALUES (1, 'Beverages',
3	public	categories	postgres	2022-06-16 08:5...	D	(1,Beverages,'Soft drinks, coffees, teas, beers, a...	[null]	DELETE FROM categories WHERE category_id = '
4	public	categories	postgres	2022-06-16 08:5...	U	(2,Condiments,'Sweet and savory sauces, relish...	(2,Condiments,condiments,'')	DELETE FROM categories WHERE category_id = '
5	public	categories	postgres	2022-06-16 08:5...	U	(2,Condiments,condiments,'')	(2,Condiments,'condiments 2','')	UPDATE categories SET description = 'condiment

Se puede ver que en esta tabla se almacenan todos los datos relevantes para realizar la auditoria, datos como, la tabla, la acción, los datos que cambiaron y cuando cambiaron. Con esto se puede tener un registro detallado de que se realiza en la base de datos en cada momento.

Bibliografía:

Flores, J. R. (2018). *Auditoria de tablas en PostgreSQL – I*. Usuario Peru TI.

<https://usuarioperu.com/2018/07/23/auditoria-de-tablas-en-postgresql-i/>

Remigio Huarcaya Almeyda. *como crear auditoria en sql server*. (2019). [Vídeo].

YouTube. <https://www.youtube.com/watch?v=nUJr5Q7W-1k>