CREATE VIEW vwVisao1

AS

select st.ID, st.name, st.tot\_cred, dep.dept\_name AS "DEPARTMENT NAME" FROM student st

JOIN department dep on dep.dept\_name = st.dept\_name

WHERE st.tot\_cred >= 50 AND st.dept\_name = 'Biology'

GO

CREATE TRIGGER Insercao\_vwVisao1

ON vwVisao1

INSTEAD OF INSERT

AS

BEGIN

INSERT INTO student(ID, name, tot\_cred, dept\_name) SELECT id, name, tot\_cred, [DEPARTMENT NAME] FROM inserted

END

GO

--Teste de inserção na VIEW vwVisao1

insert into vwVisao1 values(00127,'Alexandre', 120, 'Biology');

insert into vwVisao1 values(00450,'Paulo', 56, 'Biology');

insert into vwVisao1 values(00820,'Carlos', 78, 'Biology');

insert into vwVisao1 values(14850,'Richard', 99, 'Biology');

--Teste em que essa inserção o Aluno tem o tot\_cred abaixo de 50 logo ele não deve aparecer na View.

insert into vwVisao1 values(14859,'Colombo', 45, 'Biology');

CREATE TRIGGER delete\_vwVisao1

ON vwVisao1

INSTEAD OF DELETE

AS

BEGIN

DELETE vwVisao1 where name in (SELECT name from deleted);

END;

GO

--Teste de Exclusão do Estudante da VIEW vwVisao1

delete from vwVisao1 where name = 'Richard';

CREATE TRIGGER update\_vwVisao1

ON vwVisao1

INSTEAD OF UPDATE

AS

BEGIN

IF(UPDATE(ID))

BEGIN

RAISERROR ('ID não pode ser modificado',16,1)

RETURN

END

IF(UPDATE([DEPARTMENT NAME]))

BEGIN

RAISERROR ('[DEPARTMENT NAME] não pode ser modificado',16,1)

RETURN

END

IF(UPDATE(name))

BEGIN

DECLARE @NAME varchar(20), @ID INT;

SELECT @ID = st.ID FROM student st JOIN inserted ON inserted.ID = st.ID

IF(@ID IS NULL)

BEGIN

RAISERROR ('Aluno Inexistente!',16,1)

RETURN

END

SELECT @NAME = inserted.name FROM inserted

UPDATE student set name = @NAME FROM inserted JOIN student

ON student.ID = inserted.ID

END

END

GO

--Alterando o nome do Aluno pelo ID

UPDATE vwVisao1 set name = 'Richard Alexandre'

WHERE ID = 127;

--Validação da não alteração do ID

UPDATE vwVisao1 set ID = 15945

WHERE ID = 127;

--Validação da não alteração do [DEPARTMENT NAME]

UPDATE vwVisao1 set [DEPARTMENT NAME] = 'Biology5'

WHERE ID = 127;

--Usuários e autorizações:

CREATE LOGIN usuario1

WITH PASSWORD = 'usuario1';

GO

CREATE USER usuario1 FOR LOGIN usuario1;

GO

GRANT INSERT, UPDATE, DELETE ON dbo.student to usuario1

GRANT INSERT, UPDATE, DELETE ON dbo.department to usuario1

GRANT INSERT, UPDATE, DELETE ON dbo.course to usuario1

GRANT SELECT ON dbo.student to usuario1

GRANT SELECT ON dbo.department to usuario1

GRANT SELECT ON dbo.course to usuario1

-- Esse Comando REVOKE remove a permissão de SELECT, INSERT, UPDATE e DELETE a student para o Usuario1 USE [TB UNIVERSIDADE];

REVOKE INSERT, UPDATE, DELETE ON dbo.student to usuario1

REVOKE SELECT ON OBJECT:: dbo.student FROM usuario1;

GO

CREATE LOGIN usuario2

WITH PASSWORD = 'usuario2';

GO

CREATE USER usuario2 FOR LOGIN usuario2;

GO

GRANT SELECT ON dbo.vwVisao1 to usuario2

GRANT SELECT ON dbo.vwVisao2 to usuario2

GRANT SELECT ON dbo.vwVisao3 to usuario2

--Esse Comando REVOKE remove a permissão de SELECT a vwVisao1 para o Uusuario2

REVOKE SELECT ON OBJECT:: vwVisao1 FROM usuario2;

CREATE LOGIN usuario3

WITH PASSWORD = 'usuario3';

GO

CREATE USER usuario3 FOR LOGIN usuario3;

GO

--Permissões de procedimento armazenado

GRANT EXECUTE to usuario3

--Revogando a permissão EXECUTE para usuario1 USE [TB UNIVERSIDADE]; REVOKE EXECUTE to usuario3 GO

CREATE VIEW vwVisao2

AS

SELECT c.course\_id, c.dept\_name FROM course c

JOIN department dep ON dep.dept\_name = c.dept\_name

WHERE c.credits = 3 AND dep.building = 'Taylor'

GROUP BY c.course\_id, c.dept\_name

GO

CREATE TRIGGER Insercao\_vwVisao2

ON vwVisao2

INSTEAD OF INSERT

AS

BEGIN

DECLARE

@DEPT\_NAME varchar(20),

@COURSE\_ID varchar(8);

SELECT @DEPT\_NAME = dbo.fn\_Insert\_Department\_dept\_name(i.dept\_name)FROM inserted i; IF(@DEPT\_NAME IS NOT NULL)

BEGIN

INSERT INTO department (dept\_name, building)

SELECT @DEPT\_NAME, 'Taylor' FROM inserted

END

IF (@DEPT\_NAME IS NULL)

BEGIN

RAISERROR ('Department inserido já existe na tabela Department!',16,1)

RETURN

END

BEGIN

SELECT @COURSE\_ID = dbo.fn\_Insert\_Course\_course\_id(i.course\_id)FROM inserted i;

IF (@COURSE\_ID IS NOT NULL)

BEGIN

INSERT INTO course(course\_id, dept\_name, credits)

SELECT course\_id, dept\_name, 3 FROM inserted

END

IF (@COURSE\_ID IS NULL)

BEGIN

RAISERROR ('ID do Curso existente !',16,1)

--Como foi inserido um ID existente em course deve-se desfazer a inserção em

Department

DELETE FROM department WHERE dept\_name = @DEPT\_NAME;

RETURN

END

END

END

GO

CREATE FUNCTION fn\_Insert\_Department\_dept\_name(@DEPT\_NAME varchar(20))

RETURNS varchar(20)

BEGIN

IF((SELECT dept\_name FROM department WHERE dept\_name = @DEPT\_NAME) IS NULL)

BEGIN

RETURN @DEPT\_NAME

END

ELSE

SET @DEPT\_NAME = NULL

RETURN @DEPT\_NAME

END

CREATE FUNCTION fn\_Insert\_Course\_course\_id (@COURSE\_ID varchar(8))

RETURNS varchar(8)

BEGIN

IF((SELECT course\_id FROM course WHERE course\_id = @COURSE\_ID) IS NULL)

BEGIN

RETURN @COURSE\_ID

END

ELSE

SET @COURSE\_ID = NULL

RETURN @COURSE\_ID

END

--Inserindo valores na visão vwVisao2

INSERT INTO vwVisao2 VALUES ('MAT-101', 'Mathematics');

--Teste de inserção de valor na visão com atributo na tabela Department já existente

INSERT INTO vwVisao2 VALUES ('MAT-102', 'Mathematics');

--Teste de inserção de valor na visão com atributo na tabela Course já existente

INSERT INTO vwVisao2 VALUES ('MAT-101', 'Mathematics-1A');

--DELETE

CREATE TRIGGER delete\_vwVisao2

ON vwVisao2

INSTEAD OF DELETE

AS

BEGIN

DECLARE @COURSE\_ID varchar(8);

SELECT @COURSE\_ID = dbo.fn\_validar\_couse\_id\_FOR\_DELETE(i.course\_id) FROM deleted i;

IF(@COURSE\_ID IS NULL )

BEGIN

RAISERROR ('ID do curso inexistentes!', 16, 1)

RETURN

END

ELSE

DELETE department WHERE dept\_name in (SELECT dept\_name from deleted);

DELETE course WHERE course\_id in (SELECT course\_id from deleted);

END;

GO

CREATE FUNCTION fn\_validar\_couse\_id\_FOR\_DELETE(@COUSE\_ID varchar (8))

RETURNS varchar(8)

BEGIN

IF((SELECT course\_id FROM course WHERE course\_id = @COUSE\_ID) IS NOT NULL)

BEGIN

RETURN @COUSE\_ID

END

ELSE

SET @COUSE\_ID = NULL

RETURN @COUSE\_ID

END

--UPDATE

CREATE TRIGGER update\_vwVisao2

ON vwVisao2

INSTEAD OF UPDATE

AS

BEGIN

DECLARE @DEPT\_NAME varchar(20), @COURSE\_ID varchar(8);

IF UPDATE(course\_id)

BEGIN

SELECT @DEPT\_NAME = dep.dept\_name FROM department dep

JOIN inserted ON inserted.dept\_name = dep.dept\_name

IF(@DEPT\_NAME IS NULL)

BEGIN

RAISERROR('Nome do departamento informado inexistente!',16,1)

RETURN

END

ELSE

SELECT @COURSE\_ID = inserted.course\_id FROM inserted

UPDATE course set course\_id = @COURSE\_ID

FROM inserted JOIN course

ON course.dept\_name = inserted.dept\_name

END

END

GO

--Testando a alteração no ID do Curso

UPDATE vwVisao2 set course\_id ='MAT-102'

WHERE dept\_name = 'Mathematics';

--Testando a alteração em um departamento inexistente

UPDATE vwVisao2 set course\_id ='MAT-102'

WHERE dept\_name = 'Mathematics-A2';

Referências

<https://docs.microsoft.com/pt-br/sql/t-sql/statements/revoke-object-permissions-transact-sql?view=sql-server-2017>

<https://docs.microsoft.com/pt-br/sql/t-sql/statements/grant-object-permissions-transact-sql?view=sql-server-2017>

<https://www.devmedia.com.br/triggers-no-sql-server-teoria-e-pratica-aplicada-em-uma-situacao-real/28194>

<https://docs.microsoft.com/pt-br/sql/relational-databases/triggers/use-the-inserted-and-deleted-tables?view=sql-server-2017>

<http://codingsight.com/sql-server-triggers-understanding-alternatives/>