Views, Stored Procedures e Triggers, SQL Injection

Vitor Vaz da Silva

View

É uma expressão SQL, a composição de uma tabela, que é guardada na base de dados com um nome associado.

É útil para:

- Mostrar dados de um modo mais intuitivo para o utilizador
- Restringir o acesso a dados de modo a o utilizador aceder apenas a uma parte
- Mostrar um resumo de diversas tabelas, por exemplo para relatórios
- As views são apenas de leitura mas pode criar-se um trigger que dispara quando se tenta realizar um DELETE, INSERT, or UPDATE nessa view.

Stored Procedure

É uma sequência de comandos SQL que ficam no servidor da base de dados com o intuito de:

- ser chamada diversas vezes; uma ferramenta
- um processo com alguma segurança por não estar do lado do utilizador
- minimizar a quantidade de informação ou tarefas a serem executadas pelo utilizador no SGBD
- criar uma barreira entre os dados e o acesso aos mesmos

Trigger

É uma stored procedure que é executada aquando uma determinada acção sobre a base de dados. *(devido a um Insert, Update or Delete)*.

Está agregado a uma tabela e só pode ser executado automaticamente e nunca explicitamente.

View

```
Create View AAA(a,b,c) as
        Select …. Join …. Join … Union Select ….

-- Não é uma nova tabela, mas utiliza-se como se fosse
-- Sempre que se faz um Select … from AAA …, é executado todo o select associado ao View
-- Funciona como um alias
```


MS SQL SERVER

- System Stored Procedures
  - começam por sp_ (na base de dados master)
- User Stored Procedures –
  - começam por sp_ (na base de dados não master)
  - Podem ser:
    - User stored procedures
    - Triggers
    - User defined functions

MS SQL SERVER

Stored Procedure Syntax

```
Create  PROCEDURE <nome>
        (
        @<variavel> <tipo> [OUT] {, @<variavel> <tipo> [OUT] }*
        )
 AS BEGIN
 -- expressões SQL
END
```


MS SQL SERVER

Stored Procedure Exemplo

```
Create  PROCEDURE NomeCliente(@num INT, @nome VARCHAR(90) OUT)
AS BEGIN
 SELECT @nome= nome||' '||sobrenome
        FROM Cliente WHERE numero=@num
END
```


MS SQL SERVER

User defined function

```
Create  FUNCTION <nome>
        (
        @<variavel> <tipo> [OUT] {, @<variavel> <tipo> [OUT] }*
       )
RETURNS <tipo>
 AS BEGIN
 -- expressões SQL
 RETURN <expressão>
END
```

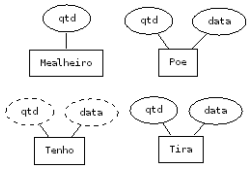

MS SQL SERVER

User defined function

```
Create  FUNCTION NomeCliente(@num INT, @nome VARCHAR(90) )
RETURNS VARCHAR(90)
AS BEGIN
 SELECT @nome= nome||' '||sobrenome
        FROM Cliente WHERE numero=@num
 RETURN @nome
END
```


Exemplo

- Tenho um mealheiro que quero manter actualizado com uma base de dados. Cada vez que tiro ou ponho alguma quantia deve ficar registada a data e hora em que isso aconteceu.
- Quero saber o valor actual e a data do último movimento.
- Devo poder ver facilmente o valor actual.
- Pode acontecer que por vezes tire ou coloque alguma quantia sem que actualize na base de dados…
- De vez em quando conto o que tenho, e nessa altura quero que a base de dados faça os acertos



Mealheiro(qtd)
Poe(qtd, data) cc={data} cp={data}
Tira(qtd, data) cc={data} cp={data}
Tenho(qtd, data)

MS SQL SERVER

```
-- Cada vez que tiro ou ponho alguma quantia deve ficar
-- registada a data e hora em que isso aconteceu.
          -- ------------------------MS SQL------------------------------- --
create table poe(
 dia datetime default CURRENT_TIMESTAMP,
 qtd float
);

create table tira(
 dia datetime default CURRENT_TIMESTAMP  ,
 qtd float);
```



```
-- Quero saber o valor actual e a data do último movimento.

      -- ------------------------MS SQL------------------------------- --

create VIEW tenho(quantia, quando) as
  select sum(total), max(dia) from(
      select sum(qtd) as total, max(dia) as dia from poe
    union
      select -sum(qtd) as total,max(dia) as dia from tira
   ) as soma
;
```



```
-- Devo poder ver facilmente o valor actual.
          -- ------------------------MS SQL------------------------------- --
create table mealheiro(  valor float default 0  );

insert into mealheiro(valor) values(0);

CREATE TRIGGER depoisDePor ON poe AFTER INSERT, UPDATE  AS
  UPDATE mealheiro set valor = valor + (SELECT qtd FROM INSERTED)
;

CREATE TRIGGER depoisDeTirar ON tira AFTER INSERT,UPDATE AS
  UPDATE mealheiro set valor = valor - (SELECT qtd FROM INSERTED)
;
```



```
-- De vez em quando conto o que tenho, e nessa altura quero
que a base de dados faça os acertos
       -- ------------------------MS SQL------------------------------- --
create procedure acerta @qtd float AS
BEGIN
 insert into poe(qtd)
        select(@qtd-(select quantia from tenho))
                where (select quantia from tenho)<@qtd
 insert into tira(qtd)
        select(-@qtd+(select quantia from tenho))
                where (select quantia from tenho)>@qtd
 update mealheiro set valor = @qtd
END;      --  exec acerta 26;
```


Experimentar

```
insert into poe(qtd) values (2.3);
insert into tira(qtd) values (0.3);
      select * from tenho;
      select * from mealheiro;

update mealheiro set valor=15;
      select * from tenho;
      select * from mealheiro;

insert into poe(qtd) values (2.3);
insert into tira(qtd) values (0.3);
      select * from tenho;
      select * from mealheiro;

exec acerta 8.20; -- Stored procedure
      select * from tenho;
      select * from mealheiro;
```

MySQL

-- Cada vez que tiro ou ponho alguma quantia deve ficar
-- registada a data e hora em que isso aconteceu.
-- ----------MySQL---------- --

```
create table poe(
   dia datetime default CURRENT_TIMESTAMP,
   qtd float)  engine=InnoDb;

create table tira(
   dia datetime default CURRENT_TIMESTAMP ,
   qtd float)  engine=InnoDb;
```


-- Quero saber o valor actual e a data do último movimento.

-- ----------MySQL---------- --

The SELECT statement cannot contain a subquery in the FROM clause.

```
create view soma(quantia, dia) as
   select sum(qtd) as total, max(dia) as dia from poe as A
   union
   select -sum(qtd) as total,max(dia) as dia from tira as B
;

create view tenho(quantia, quando) as
  select sum(quantia) as quantia, max(dia) as quando from soma
;
```


-- Devo poder ver facilmente o valor actual.

-- ----------MySQL---------- --

```
create table mealheiro( valor float default 0 ) engine=InnoDb;

insert into mealheiro(valor) values(0);

CREATE TRIGGER depoisDePor AFTER INSERT ON poe FOR EACH ROW
  UPDATE mealheiro set valor = valor + (new.qtd)
;

CREATE TRIGGER depoisDeTirar AFTER INSERT ON tira FOR EACH ROW
  UPDATE mealheiro set valor = valor -  (new.qtd)
;
```


-- De vez em quando conto o que tenho, e nessa altura quero que a base de dados faça os acertos

-- ----------MySQL---------- --

```
DELIMITER //
create procedure acerta (in qtd float)
BEGIN
   insert into poe(qtd) select * from
         (select qtd-A.quantia from (select quantia from tenho) as A)  as B
                  where (select quantia from tenho)<qtd;
   insert into tira(qtd) select * from
         (select -qtd+A.quantia from (select quantia from tenho) as A)  as B
                  where (select quantia from tenho)>qtd;
   UPDATE mealheiro set valor = qtd;
END //
DELIMITER ;  -- call acerta (26);
```


Experimentar

```
insert into poe(qtd) values (2.3);
insert into tira(qtd) values (0.3);
      select * from tenho;
      select * from mealheiro;

update mealheiro set valor=15;
      select * from tenho;
      select * from mealheiro;

insert into poe(qtd) values (2.3);
insert into tira(qtd) values (0.3);
      select * from tenho;
      select * from mealheiro;

call acerta (8.20); -- Stored procedure
      select * from tenho;
      select * from mealheiro;
```

SqLite

-- Cada vez que tiro ou ponho alguma quantia deve ficar
-- registada a data e hora em que isso aconteceu.
-- ----------SqLite---------- --

```
create table poe(
   dia datetime default CURRENT_TIMESTAMP,
   qtd float);

create table tira(
   dia datetime default CURRENT_TIMESTAMP ,
   qtd float);
```


-- Quero saber o valor actual e a data do último movimento.
-- ----------SqLite---------- --

```
create view tenho(quantia, dia) as
  select sum(total) as quantia, max(dia) as quando from (
            select sum(qtd) as total, max(dia) as dia from poe as A
       union
            select -sum(qtd) as total,max(dia) as dia from tira as B
   ) as soma
;
```


-- Devo poder ver facilmente o valor actual.
-- ----------SqLite---------- --

```
create table mealheiro( valor float default 0 );

insert into mealheiro(valor) values(0);

CREATE TRIGGER depoisDePor AFTER INSERT ON poe FOR EACH ROW
BEGIN
 UPDATE mealheiro set valor = valor + (new.qtd);
END
;

CREATE TRIGGER depoisDeTirar AFTER INSERT ON tira FOR EACH ROW
BEGIN
 UPDATE mealheiro set valor = valor -  (new.qtd);
END;
```


-- De vez em quando conto o que tenho, e nessa altura quero que a base de dados faça os acertos
-- ----------SqLite---------- --

SqLite não tem stored procedures.

Como um Trigger é uma a stored procedure, para a evocar basta escrever os parâmetros de entrada e saída numa tabela sobre a qual se faz um trigger!;

Neste caso até se podia fazer sobre a tabela Mealheiro, mas para que o exercício fique idêntico às restantes SGBD, faz-se o trigger sobre uma tabela acerta.

-- De vez em quando conto o que tenho, e nessa altura quero que a base de dados faça os acertos
-- ----------SqLite---------- --

```
create table acerta(qtd float);
insert into acerta values(0);

CREATE TRIGGER aoAcertar BEFORE UPDATE ON acerta FOR EACH ROW
BEGIN
 insert into poe(qtd) select qtd from
   (select new.qtd-A.quantia as qtd from (select quantia from tenho) as A)  as B
     where (select quantia from tenho) < new.qtd;
 insert into tira(qtd) select qtd from
    (select -new.qtd+A.quantia as qtd from (select quantia from tenho) as A)  as B
     where (select quantia from tenho) > new.qtd;
 UPDATE mealheiro set valor = new.qtd;
END;
```

Experimentar

```
insert into poe(qtd) values (2.3);
insert into tira(qtd) values (0.3);
      select * from tenho;
      select * from mealheiro;

update mealheiro set valor=15;
      select * from tenho;
      select * from mealheiro;

insert into poe(qtd) values (2.3);
insert into tira(qtd) values (0.3);
      select * from tenho;
      select * from mealheiro;

update acerta set qtd=8.20; -- "Stored procedure" as trigger
      select * from tenho;
      select * from mealheiro;
```


PostgreSQL

```
-- Cada vez que tiro ou ponho alguma quantia deve ficar
-- registada a data e hora em que isso aconteceu.
          -- ----------PostgreSQL 9.3---------- --
create table poe(
  dia timestamp DEFAULT NOW(),
  qtd float
);

create table tira(
  dia timestamp DEFAULT NOW(),
  qtd float
);
```



```
-- Quero saber o valor actual e a data do último movimento.
    -- ----------PostgreSQL 9.3 ---------- --

create view tenho(quantia, dia) as
  select sum(total) as quantia, max(dia) as quando from (
        select sum(qtd) as total, max(dia) as dia from poe as A
      union
        select -sum(qtd) as total,max(dia) as dia from tira as B
   ) as soma
;
```



```
-- Devo poder ver facilmente o valor actual.
          -- ----------PostgreSQL 9.3 ---------- --
create table mealheiro( valor float default 0 );
insert into mealheiro(valor) values(0);

create function actualizaDepoisDePor() RETURNS trigger as $depoisDePor$
BEGIN
UPDATE mealheiro set valor = valor + (NEW.qtd);
RETURN NEW;
END;
$depoisDePor$ LANGUAGE plpgsql;

CREATE TRIGGER depoisDePor BEFORE insert or update ON poe
   FOR EACH ROW EXECUTE PROCEDURE actualizaDepoisDePor();
```



```
-- Devo poder ver facilmente o valor actual. -- continua
          -- ----------PostgreSQL 9.3 ---------- --

create function depoisDeTirar() RETURNS trigger as $depoisDeTirar$
BEGIN
UPDATE mealheiro set valor = valor - (NEW.qtd);
RETURN NEW;
END;
$depoisDeTirar$ LANGUAGE plpgsql;

CREATE TRIGGER depoisDeTirar BEFORE insert or update ON tira
   FOR EACH ROW EXECUTE PROCEDURE depoisDeTirar();
```

```
-- De vez em quando conto o que tenho, e nessa altura quero que a base de dados faça os acertos
-- ---------- PostgreSQL 9.3- ----------

create function acerta(in qdtAct float) RETURNS void as $$
 BEGIN
    insert into poe(qtd) select * from
          (select qdtAct-A.quantia from (select quantia from tenho) as A)  as B
                where (select quantia from tenho)<qdtAct;
    insert into tira(qtd) select * from
          (select -qdtAct+A.quantia from (select quantia from tenho) as A)  as B
                where (select quantia from tenho)>qdtAct;
    UPDATE mealheiro set valor = qdtAct;
 END;
$$ LANGUAGE plpgsql;
```


Experimentar

```
insert into poe(qtd) values (2.3);
insert into tira(qtd) values (0.3);
      select * from tenho;
      select * from mealheiro;

update mealheiro set valor=15;
      select * from tenho;
      select * from mealheiro;

insert into poe(qtd) values (2.3);
insert into tira(qtd) values (0.3);
      select * from tenho;
      select * from mealheiro;

select acerta(8.20); -- Stored procedure
      select * from tenho;
      select * from mealheiro;
```


SQL INJECTION

A arte de colocar comandos SQL onde não é suposto serem entendidos como comandos SQL.

Nome: a'; select * from Segredo; select 'obrigado

Resposta: qualquer haha!' or 'a'='a


```
String sql;

String nome=getNome();

sql = "select nome from Pessoa where nome='"+nome+"';";

sql = String.format("select nome from Pessoa where nome='%s';", nome);
```


Prepared Statement

```
String sql ="SELECT a, b FROM Tabela WHERE a = '?' AND c = ?";

PreparedStatement stmt = conn.preparedStatement(sql);

stmt.setString(1, "um texto bonito"); -- filtrar as Strings (1=> primeiro ?)
stmt.setInt(2, 1123); -- filtrar os dígitos (2=> segundo ?)

ResultSet rs = stmt.executeQuery();
```


Referências

https://www.postgresql.org/docs/current/plpgsql.html

https://www.cisco.com/c/en/us/about/security-center/sql-injection.html