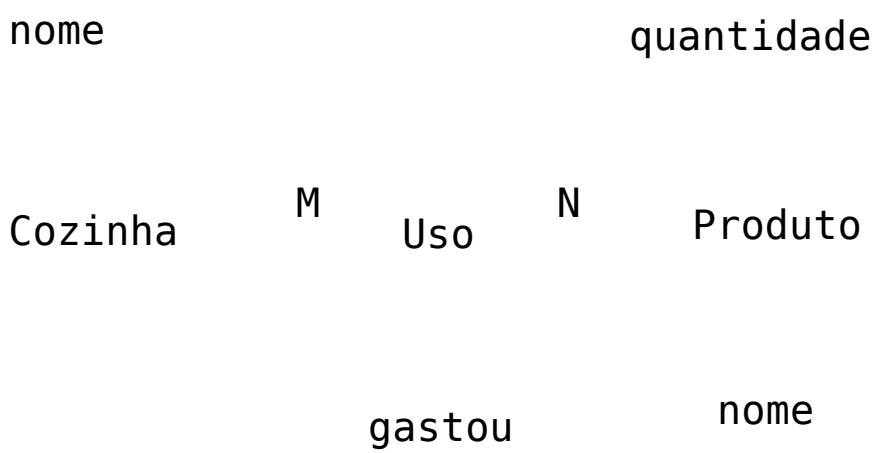
Transacções

Vitor Vaz da Silva



Em produto *nome* é único
A *quantidade* é decrescida do mesmo valor que acresce a *gastou*

MS SQL SERVER

```
create table Cozinha(
  id int IDENTITY(5,2),
  nome varchar(20),
  primary key(id)
)

create table Produto (
  id int IDENTITY(1,1),
  nome varchar(20) unique,
  quantidade int,
 primary key(id)
);
```

```
create table Uso (
  cli int,
  prod int,
  gastou int,
  foreign key (cli) references Cozinha(id),
  foreign key (prod) references Produto(id),
  primary key(cli,prod)
);
```

```
insert into Cozinha(nome) values ('Ana'),('Bruno'),('Carla'),('David');

insert into Produto(nome,quantidade) values
('Salsa',10),('Manjericão',20),('Cardamono',30),('Curcuma',40);
```

MySQL

```
create table Cozinha(
  id int AUTO_INCREMENT,
  nome varchar(20),
  primary key(id)
)

create table Produto (
  id int AUTO_INCREMENT,
  nome varchar(20) unique,
  quantidade int,
 primary key(id)
);
```

```
create table Uso (
  cli int,
  prod int,
  gastou int,
  foreign key (cli) references Cozinha(id),
  foreign key (prod) references Produto(id),
  primary key(cli,prod)
);
```

```
insert into Cozinha(nome) values ('Ana'),('Bruno'),('Carla'),('David');

insert into Produto(nome,quantidade) values
('Salsa',10),('Manjericão',20),('Cardamono',30),('Curcuma',40);
```

SQLite

```
create table Cozinha(
  id int AUTOINCREMENT,
  nome varchar(20),
  primary key(id)
)

create table Produto (
  id int AUTOINCREMENT,
  nome varchar(20) unique,
  quantidade int,
 primary key(id)
);
```

```
create table Uso (
  cli int,
  prod int,
  gastou int,
  foreign key (cli) references Cozinha(id),
  foreign key (prod) references Produto(id),
  primary key(cli,prod)
);
```

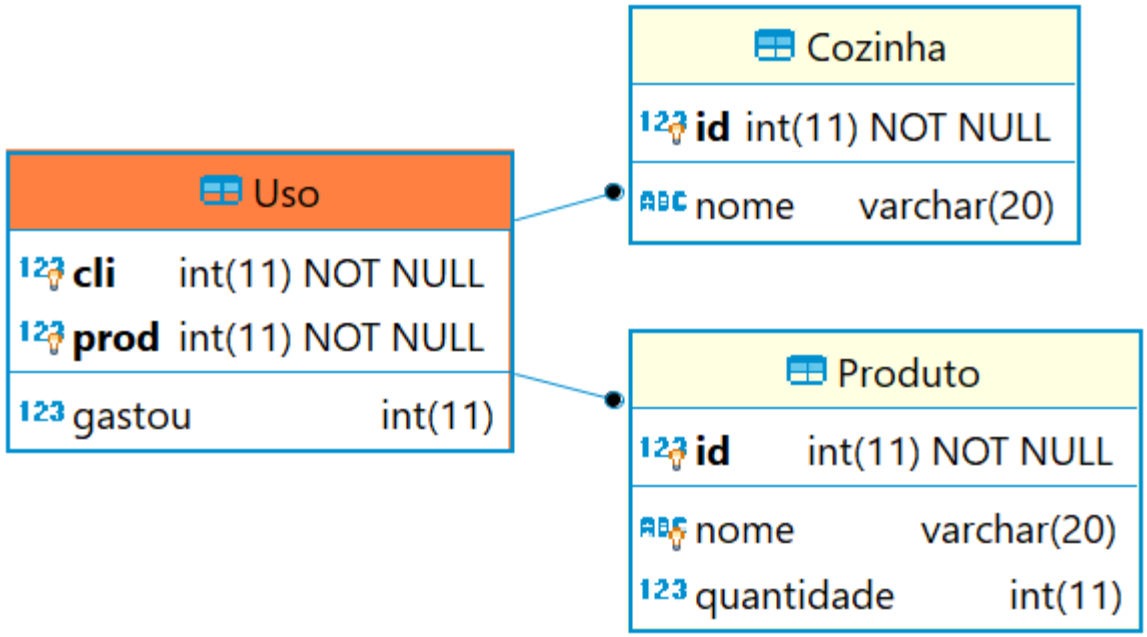
```
insert into Cozinha(nome) values ('Ana'),('Bruno'),('Carla'),('David');

insert into Produto(nome,quantidade) values
('Salsa',10),('Manjericão',20),('Cardamono',30),('Curcuma',40);
```

PostgreSQL

```
create table Cozinha(
  id int SERIAL,
  nome varchar(20),
  primary key(id)
)

create table Produto (
  id int SERIAL,
  nome varchar(20) unique,
  quantidade int,
 primary key(id)
);
```

```
create table Uso (
  cli int,
  prod int,
  gastou int,
  foreign key (cli) references Cozinha(id),
  foreign key (prod) references Produto(id),
  primary key(cli,prod)
);
```

```
insert into Cozinha(nome) values ('Ana'),('Bruno'),('Carla'),('David');

insert into Produto(nome,quantidade) values
('Salsa',10),('Manjericão',20),('Cardamono',30),('Curcuma',40);
```


Uso
cli int(11) NOT NULL
prod int(11) NOT NULL
gastou int(11)
Cozinha
id int(11) NOT NULL
nome varchar(20)
Produto
id int(11) NOT NULL
nome varchar(20)
quantidade int(11)

Exemplo

-- A Ana utilizou uma unidade de Salsa

```
UPDATE Produto
  SET
     quantidade = quantidade-1
  WHERE nome='Salsa'
;

UPDATE Uso
  SET
    gastou=gastou+1
  WHERE (cli=SELECT id FROM Cozinha WHERE nome='Ana')
      AND (prod=SELECT id FROM Produto WHERE nome='Salsa')
;
```

Actualizar duas tabelas,
E uma delas pode nem ter qualquer tuplo que possa ser actualizado!

-- Inserir 0 em quantidade se não existe entrada

-- Actualizar a quantidade com o valor usado

```
INSERT INTO Uso(cli, prod, gastou)
  select
     (select id from Cozinha where nome=‘Ana'),
     (select id from Produto where nome='Salsa'),
     0
  where not exists (
     select * from Uso
       where cli=(select id from Cozinha where nome=‘Ana')
         and prod=(select id from Produto where nome='Salsa')
   )
;
```

```
-- Actualizar a quantidade com o valor usado

UPDATE Uso
  SET
    gastou=gastou+1
  WHERE
       (cli=SELECT id FROM Cozinha WHERE nome='Ana')
     AND
       (prod=SELECT id FROM Produto WHERE nome='Salsa')
;
```

E se falhar algo ?

-- Forma mais simples.

-- Para actualizar diversas tabelas em simultâneo

-- mantendo a integridade da base de dados

BEGIN TRANSACTION

-- tudo o que for atómico

COMMIT

EXEMPLO

-- Para fazer uma salada o Bruno usou 3 unidades de Cardamono e a Carla 2 unidades de Manjericão.

MS SQL SERVER

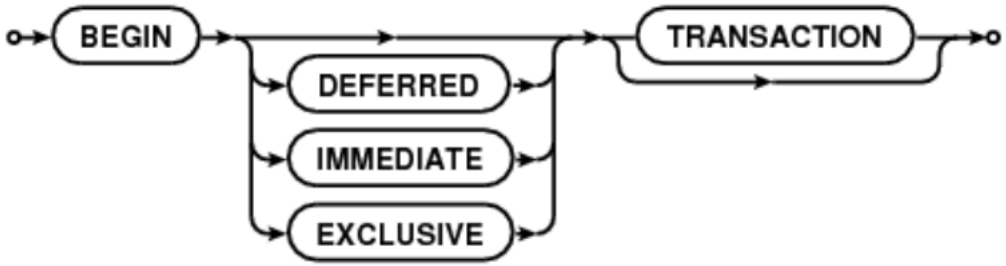
```
BEGIN TRANSACTION
    BEGIN TRY
        -- tudo o que for atómico
        commit
    END TRY
    BEGIN CATCH
        rollback;
    END CATCH;
```

MySQL

```
PROCEDURE `faz_transaccao`()
BEGIN

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
    END;

    START TRANSACTION;
        -- tudo o que for atómico
    COMMIT;

END
```

SQLite

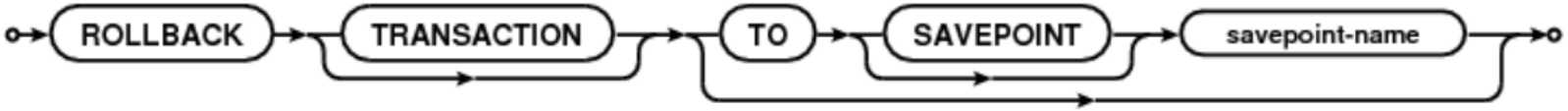
begin-stmt: hide



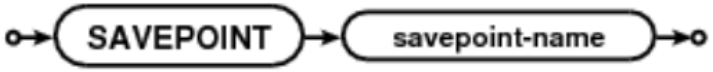
commit-stmt: hide

End é um alias para Commit

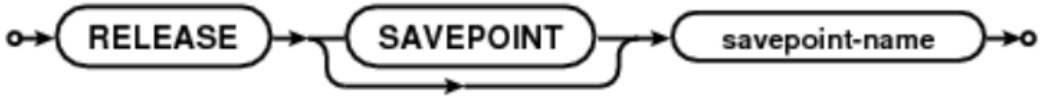
rollback-stmt: hide



savepoint-stmt: hide



release-stmt: hide



Procurar

- SAVEPOINT
- RELEASE
- Erros possíveis
- Comportmento:
  - se bloqueia
  - se permite transacções dentro de transacções