Triggers

- um trigger permite que uma determinada sequência de comandos SQL seja accionada quando um determinado evento ocorre.

- o evento pode ser INSERT, UPDATE, ou DELETE.

- o trigger pode ser accionado imediatamente antes (BEFORE) ou imediatamente depois (AFTER) de cada evento.

- o procedimento pode ser programado numa linguagem qualquer desde que devidamente instalada no servidor do SGBD.

- no exemplo que se segue, vamos utilizar PL/pgSQL, uma linguagem procedimental muito parecida com a linguagem PL/SQL da Oracle.

Exemplo

Pretende-se apagar um estúdio se não houver nenhum filme que lhe faça referência.

- associamos um trigger à tabela de filmes.
- o trigger será accionado logo após um delete ou um update à tabela de filmes.
- ao ser disparado, o trigger irá executar um procedimento chamado `apaga_estudio_sem_ref()`

```
CREATE TRIGGER estudio_sem_ref
AFTER DELETE OR UPDATE ON Filmes
FOR EACH ROW
   EXECUTE PROCEDURE apaga_estudio_sem_ref();
```

Exemplo (cont.)

- o procedimento verifica se existe algum filme produzido pelo estúdio que acabou de ser apagado (ou actualizado). Em caso negativo, apaga esse estúdio da tabela de estúdios.

```
CREATE FUNCTION apaga_estudio_sem_ref () RETURNS OPAQUE
AS '
DECLARE
   estudio VARCHAR(40);
BEGIN
   estudio := OLD.nomeEstudio;
   IF NOT EXISTS (SELECT * FROM Filmes
                  WHERE nomeEstudio = estudio)
   THEN
       DELETE FROM Estudios WHERE nome = estudio;
       RAISE NOTICE ''Estudio % foi apagado.'', estudio;
   END IF;
   RETURN NULL;
END;
' LANGUAGE 'plpgsql';
```

Rules (Regras)

- as rules (regras em português) não fazem parte do SQL standard, mas são implementadas pelo PostgreSQL e dão muito jeito.

- permitem especificar uma acção alternativa que é executada em selects, inserts, updates, ou deletes.

- sintaxe:

```
CREATE RULE name AS ON event
  TO object [WHERE rule_qualification]
  DO [INSTEAD] [ action    |
                 (actions) |
                  NOTHING
               ];
```

Rules (cont.)

- quando o evento é um insert, update, ou delete, podemos fazer referência a duas pseudo-tabelas, NEW e OLD, que se referem aos novos tuplos (no caso de inserts e updates) e aos velhos tuplos (no caso de deletes e updates).

- exemplo: a seguinte regra não permite apagar filmes da Disney.

```
CREATE RULE nao_apaga_filmes_disney AS
  ON DELETE TO Filmes
  WHERE OLD.nomeEstudio = 'Disney'
  DO INSTEAD NOTHING;
```

Rules e Views

- em PostgreSQL, as views são implementadas com rules.

- em PostgreSQL, as views são "read-only", mas podemos simular a "escrita" em views utilizando rules.

- exemplo: criar uma regra de modo a podermos inserir tuplos na view `filmes_disney`.

```
CREATE VIEW filmes_disney (nome,ano,realizador) AS
  SELECT nome, ano, nomeRealizador
  FROM Filmes
  WHERE nomeEstudio = 'Disney';
```

Exemplo (cont.)

```
CREATE RULE insere_filmes_disney AS
  ON INSERT TO filmes_disney
  DO INSTEAD
    INSERT INTO Filmes (nome, ano,
                        nomeRealizador, nomeEstudio)
    VALUES (NEW.nome, NEW.ano, NEW.realizador, 'Disney');
```

Agora já podemos inserir dados na view.

```
INSERT INTO filmes_disney
VALUES ('The Huntchback of Notre Dame',
        1995, 'Steven Spielberg');
```

Outro exemplo

- inserir dados na view `filmes_actores`

```
CREATE VIEW filmes_actores (filme, ano, estudio,
                            realizador, actor)   AS
    SELECT F.nome, F.ano, F.nomeEstudio,
           F.nomeRealizador, P.nomeActor
    FROM Filmes AS F, Participa AS P
    WHERE F.nome = P.nomeFilme
      AND F.ano = P.anoFilme;
```

- a inserção de um tuplo na view, pode fazer com que tenhamos de inserir tuplos na tabela `Filmes`, `Estudios`, `Realizadores`, `Actores` e `Participa`.

Outro exemplo (cont.)

```
CREATE RULE insere_em_filmes_actores AS
ON INSERT TO filmes_actores
DO INSTEAD
 (INSERT INTO Filmes(nome, ano,
                     nomeEstudio, nomeRealizador)
    VALUES (NEW.filme, NEW.ano,
            NEW.estudio, NEW.realizador);

  INSERT INTO Actores(nome) VALUES (NEW.actor);

  INSERT INTO Estudios(nome) VALUES (NEW.estudio);

  INSERT INTO Realizadores(nome) VALUES (NEW.realizador);

  INSERT INTO Participa(nomeFilme,anoFilme,nomeActor)
    VALUES (NEW.filme, NEW.ano, NEW.actor);
 );
```

Agora podemos inserir dados na view `filmes_actores`.

```
INSERT INTO filmes_actores
  VALUES ('Taxi Driver', 1978, 'Universal',
          'Martin Scorcese', 'Roxana Arquette');
```

- os inserts podem dar erro porque podem violar restrições que hajam nas tabelas. Por exemplo, se o estúdio `Universal` já existir na tabela de estúdios, o insert falha porque o nome do estúdio é chave primária.

- a solução para este problema é criar regras adicionais. Por exemplo:

```
CREATE RULE nao_insere_estudios_duplicados AS
  ON INSERT TO Estudios
  WHERE( EXISTS
           (SELECT * FROM Estudios
            WHERE nome=NEW.nome
           )
       )
  DO INSTEAD NOTHING;
```

Outro exemplo (cont.)

- a inserção de dados na view "esconde" a inserção efectiva dos dados nas tabela base.

- ao inserir dados na view ficamos impossibilitados de inserir todos os atributos das tabelas base.

- esses atributos ficam com o valor NULL (ou com o valor que estiver especificado como DEFAULT).

- podem consultar mais coisas na documentação online (Programmer's Guide → Server Programming → The Postgres Rule System)