Restrições de Integridade Semântica

- Controle de valores válidos para os dados
  - estados dos dados condizentes com os requisitos da aplicação
  - transições de estados corretas

- O SGBD deve controlar esta integridade
  - subsistema de integridade semântica

Subsistema de Integridade Semântica

- Funções básicas
  - especificação de regras de integridade (DDL)
  - monitoração das atualizações de dados
  - tratamento de violações de integridade ocasionadas por estas operações
- Definição de uma Regra de Integridade (RI)
  - para quais dados deve-se verificar a regra
  - quando a regra deve ser verificada
  - que ação deve ser tomada
- Garantia de independência de gerenciamento de integridade de dados para as aplicações

RIs em SQL

- RIs associadas à criação de tabelas
  - cláusula ***not null***
  - cláusula ***unique***
  - cláusula ***check***
  - restrições de integridade de entidade e referencial
- Assertivas *(Assertions)*: predicados que devem ser sempre verdadeiros (ativação automática)
- Gatilhos *(Triggers)*: disparo de ações vinculadas à execução de uma operação específica (ativação automática)
- Procedimentos (*Stored Procedures*): podem usadas para verificação de integridade (ativação pela aplicação ou pelo usuário)

RIs na Definição de Tabelas

```
create table Empregados (

codEmp integer,
nome  varchar(40)not null,
RG   numeric(10) not null unique,
idade      integer check (idade between 16 and
  90),
estadoCivil char(10) check (estado_civil in
  'solteiro', 'casado', 'viuvo', 'desquitado',
  'divorciado'),
salario numeric(8,2) check (salário > 0),
tempoServiço integer,
codGer  integer,
codDepto  integer,
...
```

RIs na Definição de Tabelas

```
...
constraint EmpPk primary key (codEmp),
constraint EmpCodg foreign key (codGer)
  references Empregados on update cascade

                          on delete set null,
constraint EmpDept foreign key (codDepto)
  references Departamentos on update cascade
                          on delete no action,
constraint IdadeTS check(tempoServico < idade),
constraint EmpGer check(codEmp < > codGer),
constraint SalarioGerente
     check(salario < (select salario
                      from Empregados e
                      where e.codEmp = codGer))
 );
```

RIs na Definição de Tabelas

```
create table Departamentos (

codDepto integer,
nome varchar(20) check (nome  in 'Vendas',
   'Pessoal', 'Finanças', 'Administrativo'),
Andar integer check (andar between 1 and 10),
orçamento numeric(20,2),

constraint DeptPk primary key (codDepto),
constraint ControleOrç
        check( orçamento > =
                  (select sum(salário)
                   from Empregados e
                   where e.codDepto = codDepto))
);
```

Assertiva - *Assertion*

- Declaração de predicado que deve ser sempre verdadeiro
  - não está vinculado a uma tabela específica
    - geralmente é uma RI que envolve várias tabelas
  - independe da operação de atualização
- Exemplo

```
create assertion ControleDespesas
check ( (select sum(ValorTotal)
          from Empréstimos)          +
          (select sum(Orçamento)
           from Departamentos)
           <= 50000000.00) )
```

  - esta afirmação é verificada mesmo se alguma das tabelas estiver vazia

Gatilho - *Trigger*

- Execução automática de modificações no BD para garantia de integridade
- Princípio de funcionamento
  - evento-[condição]-ação
- Especificação de um *trigger*
  - evento que o dispara: *comando atualização*
  - condição (opcional): *predicado*
  - ação(ões) a realizar: *comandos SQL ou execução de procedimentos (stored procedures)*

Exemplos de Gatilho

a)
```
create trigger EmpréstimoLiquidado
 after update on Empréstimos
 (delete from Empréstimos
  where valorTotal = 0)
```

b)
```
create trigger SalárioInjusto
 after insert, update on Empregados
    if exists (select *
               from Empregados
               where estadoCivil = ´casado´
               and salário < 1000.00)
    begin
        print 'salário injusto para o estado civil!'
        exec CorrigeSalário
    end
```

c)
```
create trigger AumentoSalário
 before update on Empregados
     if NEW.salario < OLD.salario
      (Rollback Transaction)
```

Procedimento – *Stored Procedure*

- Conjunto de comandos SQL mantido no BD
- Pode ser utilizado para controle de RI
  - execução a cargo da aplicação ou usuário
- Exemplo
  - débitos diários de parcelas de empréstimos

```
create procedure PagaEmpréstimo as
    update Empréstimos
    set ValorTotal = ValorTotal - ValorParcela,
        NroParcelas = NroParcelas - 1
    where DiaVencimento = day(getdate())
```

- Chamada de um *storedprocedure*

```
exec PagaEmpréstimo
```

Classificação de RIs X Suporte SQL

- RIs quanto ao alcance de dados
  - cláusula *check*; *asserts* e *triggers*
- RIs quanto ao momento da verificação
  - imediato (*default*)
  - postergado (SQL padrão)

```
set integrity for Departamentos off
...
set integrity for Departamentos immediate checked
```

- RIs de transição de estado
  - *triggers* (com referência a valores novos/antigos)
- RIs de ativação explícita
  - *stored procedures*

RIs no Postgres

- *Triggers* X *Rules*
  - *Triggers*
    - Invocam sempre uma função que trata a restrição
    - Pode ser invocado para cada tupla afetada pelo comando de atualização

```
CREATE TRIGGER SalárioInjusto BEFORE INSERT OR UPDATE ON Empregados
FOR EACH ROW EXECUTE PROCEDURE VerificaSalárioAlterado();
```

  - *Rules*
    - Similar a uma *trigger* (*before*) convencional
    - Execução adicional (`ALSO`) ou alternativa (`INSTEAD`)

```
CREATE RULE EmpréstimoLiquidado AS ON UPDATE TO Empréstimos DO ALSO
   DELETE FROM Empréstimos
   WHERE valorTotal = 0)
```

RIs no Postgres

- Triggers X cláusulas *Check* e Assertivas
  - Cláusula *check* não permite subconsultas
  - Postgres não implementa Assertvas
  - *Triggers* e funções devem ser utilizados para implementar estes tipos de RIs
  - exemplo: salário do empregado deve ser menor que o salário do gerente

```
CREATE TABLE Empregados
(...
CHECK(salario <
(select salario
 from Empregados e
 where e.codEmp = codGer)
)
```

```
CREATE TRIGGER SalárioGerente
BEFORE INSERT OR UPDATE ON
Empregados
FOR EACH ROW
EXECUTE PROCEDURE
ComparaSalarios();
```

RIs no Postgres

- RIs Postergadas
  - O comando `ALTER TABLE` permite habilitar ou desabilitar *triggers*
  - O comando `SET CONSTRAINTS` permite definir uma *trigger* ou *rule* como imediata ou postergada
    - Quando uma *trigger* ou *rule* passa de *deferred* → *immediate*, testa retroativamente as alterações já realizadas

```
...
ALTER TABLE DISABLE TRIGGER SalárioGerente
...
SET CONSTRAINTS EmpréstimoLiquidado DEFERRED
...
SET CONSTRAINTS EmpréstimoLiquidado IMMEDIATE
...
```