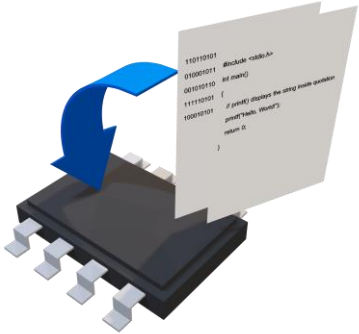
UNISINOS

DESAFIE O AMANHÃ.

BANCO DE DADOS PARA ENGENHARIA

Prof. Armando L. Keller



Restrições de integridade complexas

Restrições de integridade complexas

Além das restrições já vistas anteriormente, o SQL aceita alguns outros tipos de restrições de integridade mais complexas, como restrições de tabela, de domínio, e até a criação de tipos distintos.

Restrições de integridade complexas

As restrições de tabela, são aplicadas sobre uma única tabela, e são utilizadas na criação da tabela na forma CHECK expressão-condicional.

Restrições de integridade complexas

Um exemplo pode ser visto para validar os dados inseridos na tabela marinheiros, para garantir que a avaliação seja um número entre 1 e 10.

```
CREATE TABLE Marinheiros ( id-marin     INTEGER,
                           nome-marin   CHAR (10) ,
                           avaliação    INTEGER,
                           idade        REAL,
                           PRIMARY KEY (id-marin),
                           CHECK ( avaliação >= 1 AND avaliação <= 10 ))
```

Restrições de integridade complexas

Alguns SGBDs, suportam a criação de domínios específicos que já implementam o check por padrão, e até mesmo a criação de um noo tipo a ser passado na criação do banco de dados.

No entanto o MySQL não suporta nem a criação de domínios e nem a criação de tipos definidos pelo usuário.

Gatilhos (Triggers)

Gatilhos (Triggers)

A linguagem SQL possui um recurso chamado **gatilho** que é um procedimento que chamado sempre que um determinado evento ocorrer no banco de dados.

Um banco de dados que possui gatilhos associados pode ser chamado de banco de dados ativo.

Gatilhos (Triggers)

As partes integrantes da criação de um gatilho são

- **Evento:** A alteração no banco que ativará o gatilho;
- **Condição:** A consulta ou teste que será executado quando o gatilho for acionado pelo evento;
- **Ação:** O procedimento que será executado caso a condição determinada for verdadeira.

Gatilhos (Triggers)

A sintaxe para a criação do gatilho é

CREATE TRIGGER **nome_t momento_t evento_t**
ON **tabela_t**
FOR EACH ROW [**ordem_t**]
corpo_t

Gatilhos (Triggers)

Onde
Nome_t: Nome do gatilho a ser salvo;
Momento_t: Momento em que o gatilho irá atuar (aceita os valores BEFORE e AFTER)
Evento_t: Evento que acionará o gatilho (aceita os valores INSERT, UPDATE e DELETE)
Ordem_t: Parametro opcional, aceita os valores FOLLOWS ou PRECEDES seguido do nome de outro gatilho.
Tabela_t: Nome da tabela a ser monitorada;
Corpo_t: Corpo do trigger a ser executado.

Gatilhos (Triggers)

Exemplo: Para um sistema onde são armazenadas as informações de transações financeiras, deseja-se registrar todas as transações em uma tabela **Transacoes** que armazenará um identificador único para cada transação, além do valor. Mas deseja-se separar estes valores em duas outras tabelas **Entradas** e **Saidas** que armazenarão um identificador único para cada entrada e para cada saída, além do valor e do identificador da transação.

Gatilhos (Triggers)

Criando as tabelas

```
create table Transacoes(    id_transacao integer primary key auto_increment,
                            valor decimal);

create table Entradas(  id_entrada integer primary key auto_increment,
                        valor decimal,
                        id_transacao integer,
                        foreign key (id_transacao) references Transacoes(id_transacao));

create table Saidas (   id_saida integer primary key auto_increment,
                        valor decimal,
                        id_transacao integer,
                        foreign key (id_transacao) references Transacoes(id_transacao));
```

Gatilhos (Triggers)

Criando o gatilho

```
delimiter $$
create trigger MonitoraEntradas after insert ON Transacoes
    for each row begin
            if (new.valor> 0) then
                insert into Entradas(id_transacao,valor)
                values (new.id_transacao,new.valor);
            else
                insert into Saidas(id_transacao,valor)
                values (new.id_transacao, new.valor);
            end if;
        end $$
delimiter ;
```

Gatilhos (Triggers)

Agora deseja-se criar uma restrição para que não seja permitido inserir uma transação com valor negativo caso o saldo seja inferior ao valor a ser retirado.

Gatilhos (Triggers)

Criando o gatilho

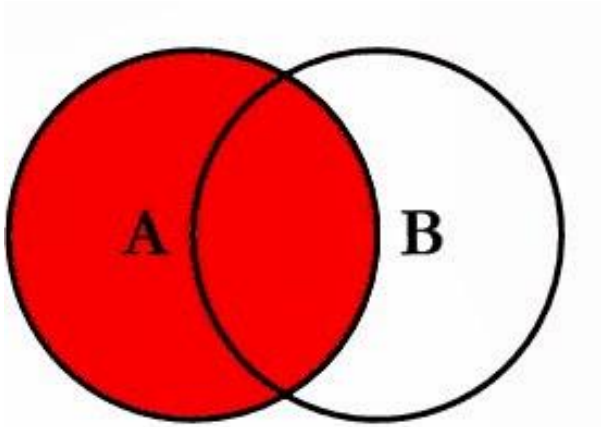
```
delimiter $$
create trigger EvitaDivida BEFORE INSERT ON Transacoes
    for each row begin
        declare saldo decimal;

        set @saldo := (SELECT sum(T.valor) from Transacoes T);
        if (@saldo+new.valor)<0 then
            signal sqlstate '45000' SET MESSAGE_TEXT = 'Saldo insuficiente' ;
        end if;
    end $$
delimiter ;
```

Junções (Joins)

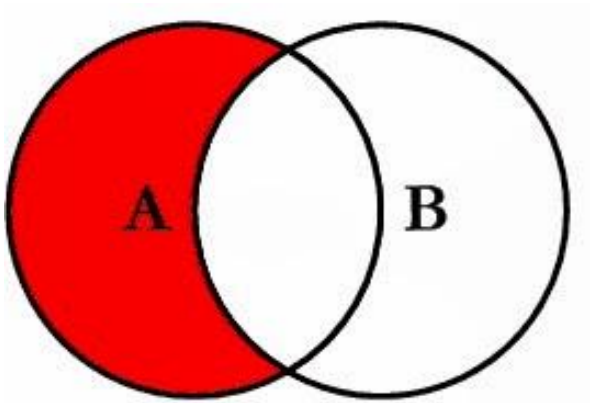
Junçoes (Joins)

Em SQL as junções são realizadas pelo comando JOIN e suas variantes, estes já executam operações de conjunto baseadas em igualdades de colunas.

Junçoes (Joins)

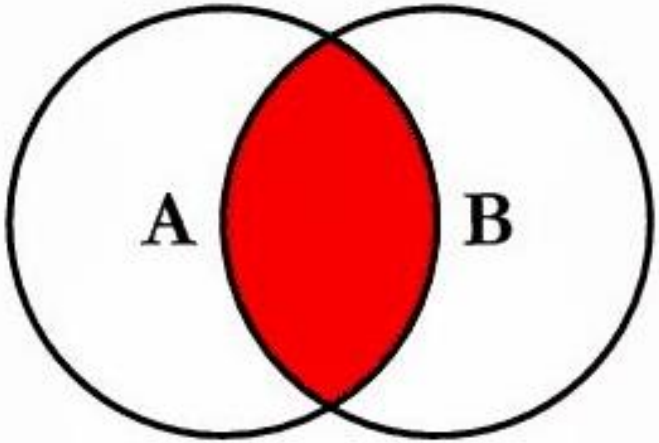


Junçoes (Joins)



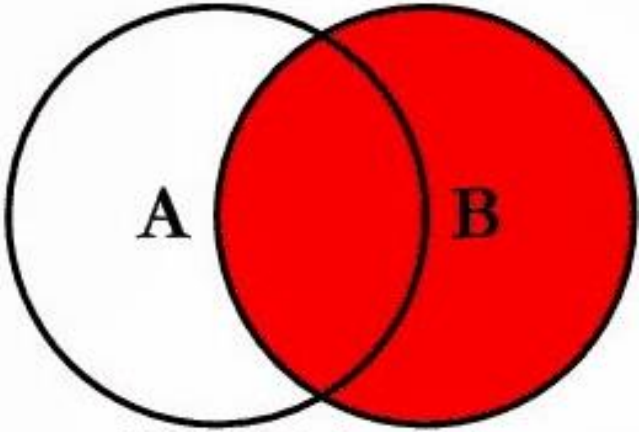

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

Junçoes (Joins)



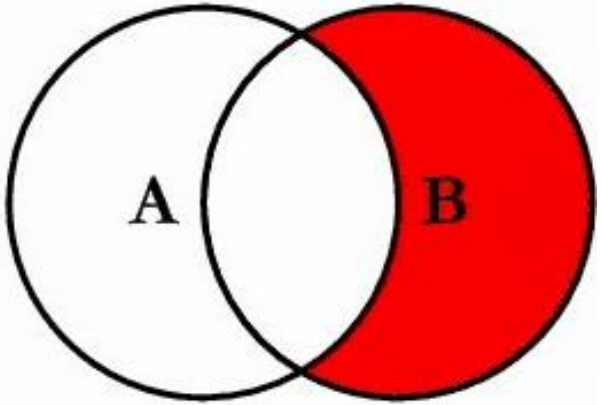

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

Junçoes (Joins)



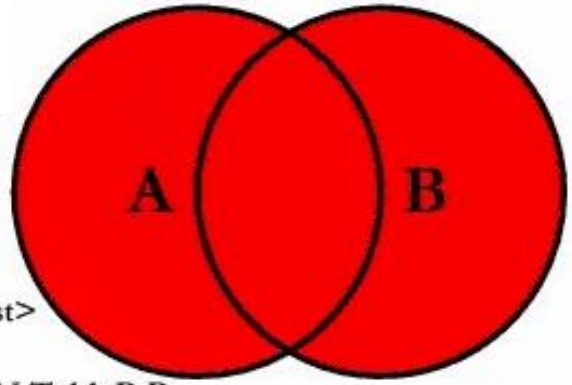

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

Junçoes (Joins)



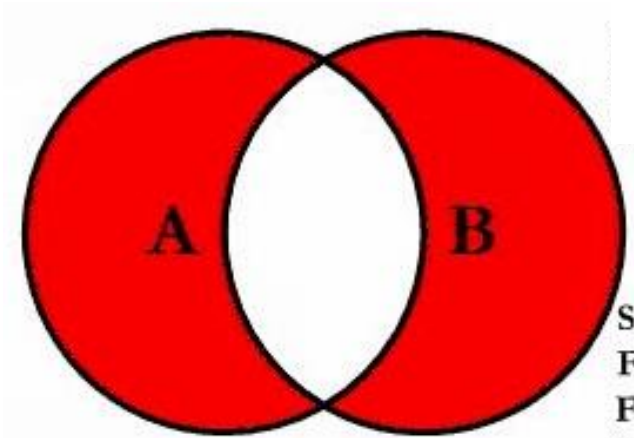

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

Junçoes (Joins)




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

Junçoes (Joins)




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Junçoes (Joins)

Além destas opções ainda pode-se utilizar o CROSS JOIN que associará cada elemento de uma tabela com cada elemento de outra tabela, mesmo que estes não compartilhem os mesmos valores.

Junçoes (Joins)

Caso a junção deva ser feita para a condição onde todas as colunas compartilhadas forem iguais, pode-se utilizar a palavra NATURAL antes do JOIN;

Por exemplo no caso dos marinheiros, pode-se fazer a junção de todos os marinheiros e reservas e barcos e obter todas as informações referente as reservas.

Junçoes (Joins)

Exemplo: No caso dos marinheiros deseja-se retornar as informações dos barcos e das reservas.
Isto pode ser feito de varias maneiras:

a) SELECT * FROM Barcos as B INNER JOIN Reservas as R ON B.id_barco = R.id_barco;

b) SELECT * FROM Barcos as B INNER JOIN Reservas as R USING(id_barco);

c) SELECT * FROM Barcos NATURAL JOIN Reservas;

Junçoes (Joins)

Caso A

```
mysql> select * from Barcos as B inner join Reservas as R ON R.id_barco=B.id_barco;
+-----------+-------------+----------+-----------+-----------+------------+
| id_barco  | nome_barco  | cor      | id_marin  | id_barco  | dia        |
+-----------+-------------+----------+-----------+-----------+------------+
|       101 | Interlake   | azul     |        22 |       101 | 1998-10-10 |
|       101 | Interlake   | azul     |        64 |       101 | 1998-05-09 |
|       102 | Interlake   | vermelho |        22 |       102 | 1998-10-10 |
|       102 | Interlake   | vermelho |        31 |       102 | 1998-10-11 |
|       102 | Interlake   | vermelho |        64 |       102 | 1998-05-09 |
|       103 | Clipper     | verde    |        22 |       103 | 1998-08-10 |
|       103 | Clipper     | verde    |        31 |       103 | 1998-06-11 |
|       103 | Clipper     | verde    |        74 |       103 | 1998-08-09 |
|       104 | Marine      | vermelho |        22 |       104 | 1998-07-10 |
|       104 | Marine      | vermelho |        31 |       104 | 1998-12-11 |
+-----------+-------------+----------+-----------+-----------+------------+
10 rows in set (0.01 sec)
```

Junçoes (Joins)

Caso B

```
mysql> select * from Barcos as B inner join Reservas as R using(id_barco);
+----------+------------+----------+----------+------------+
| id_barco | nome_barco | cor      | id_marin | dia        |
+----------+------------+----------+----------+------------+
|      101 | Interlake  | azul     |       22 | 1998-10-10 |
|      101 | Interlake  | azul     |       64 | 1998-05-09 |
|      102 | Interlake  | vermelho |       22 | 1998-10-10 |
|      102 | Interlake  | vermelho |       31 | 1998-10-11 |
|      102 | Interlake  | vermelho |       64 | 1998-05-09 |
|      103 | Clipper    | verde    |       22 | 1998-08-10 |
|      103 | Clipper    | verde    |       31 | 1998-06-11 |
|      103 | Clipper    | verde    |       74 | 1998-08-09 |
|      104 | Marine     | vermelho |       22 | 1998-07-10 |
|      104 | Marine     | vermelho |       31 | 1998-12-11 |
+----------+------------+----------+----------+------------+
10 rows in set (0.00 sec)
```

Junçoes (Joins)

Caso C

```
mysql> SELECT * FROM Barcos NATURAL JOIN Reservas;
+----------+------------+----------+----------+------------+
| id_barco | nome_barco | cor      | id_marin | dia        |
+----------+------------+----------+----------+------------+
|      101 | Interlake  | azul     |       22 | 1998-10-10 |
|      101 | Interlake  | azul     |       64 | 1998-05-09 |
|      102 | Interlake  | vermelho |       22 | 1998-10-10 |
|      102 | Interlake  | vermelho |       31 | 1998-10-11 |
|      102 | Interlake  | vermelho |       64 | 1998-05-09 |
|      103 | Clipper    | verde    |       22 | 1998-08-10 |
|      103 | Clipper    | verde    |       31 | 1998-06-11 |
|      103 | Clipper    | verde    |       74 | 1998-08-09 |
|      104 | Marine     | vermelho |       22 | 1998-07-10 |
|      104 | Marine     | vermelho |       31 | 1998-12-11 |
+----------+------------+----------+----------+------------+
10 rows in set (0.00 sec)
```

CASE

CASE

Outro comando que potencializa o poder das consultas em SQL é o CASE, que pode ser utilizado para avaliar varias condições para então retornar um valor.

CASE

Sintaxe:

```
CASE
        WHEN condição 1 THEN resultado 1
        WHEN condição 2 THEN resultado 2
        ELSE  resultado 3
END;
```

CASE

Exemplo: Considerando que agora reservas possui um campo dia_devolucao, indicar se o barco foi devolvido para cada reserva.

```
SELECT R.id_barco,
CASE
   WHEN R.dia_devolucao THEN 1
   ELSE 0
END as devolvido
FROM Reservas R;
```

CASE

id_barco	devolvido
101	1
102	1
103	1
104	1
102	1
103	1
104	0
101	0
102	0
103	0

Exercícios

Exercício 1: Considerando o banco de dados do caso Marinheiros

a) Modifique as tabelas para que seja informada a data de devolução do barco;

b) Modifique as tabelas para que os barcos possuam Categorias

A – Para qualquer nível de avaliação

B – Somente para marinheiros com avaliação superior a 6

C – Somente para marinheiros com avaliação superior a 9

Exercícios

d) Crie um gatilho que evite que um marinheiro com avaliação inferior a categoria do barco faça a reserva;

e) Crie um gatilho para que um barco que ainda não foi devolvido não possa ser reservado;

Exercícios

Exercício 2: Para o caso dos Marinheiros, crie uma visão que retorne um resumo das reservas, utilizando junções, onde deve ser apresentado:
Id_marin, nome_marin, id_barco, nome_barco e tempo de reserva

OBRIGADO.

UNISINOS

DESAFIE
O AMANHÃ.