Restrições Integridade em SQL

Bancos de Dados I
Altigran Soares da Silva
IComp/UFAM - 2016/02
Adaptado dos Slides do Professor Jeffrey Ullman

Esquema de uma Relação

- Esquema de uma relação:
  - Nome
  - Atributos com os respectivos domínios
  - Chaves
- Linguagem de Definição de Dados
  - Usada para descrever o esquema
  - A SQL é em parte uma LDD
- Armazenado como *meta-dados* no catálogo
- Consistência:
  - Instâncias "respeitam" o esquema

Restrições de Integridade - Nativas

- Restrição de Domínio
  - Todo atributo só assume valores de seu domínio
- Restrição de Chave
  - Não existem duas tuplas com o mesmo valor para uma chave
- Restrição de Entidade
  - Chaves primárias não pode assumir NULL
- Restrição de Valores Nulos
  - Garantia do "NOT NULL"

Chave Atributo Simples

- PRIMARY KEY ou UNIQUE após a declaração do domínio do atributo.
- Exemplo:

```
CREATE TABLE Cervejas (
        nome  CHAR(20) UNIQUE,
        fabr  CHAR(20)
);
```

Chave multi-atributo

- Os atributos bar e cerveja juntos formam a chave para a relação Vendas:

```
CREATE TABLE Vendas (
    bar      CHAR(20),
    cerveja  VARCHAR(20),
    preco    REAL,
    PRIMARY KEY (bar,cerveja)
);
```

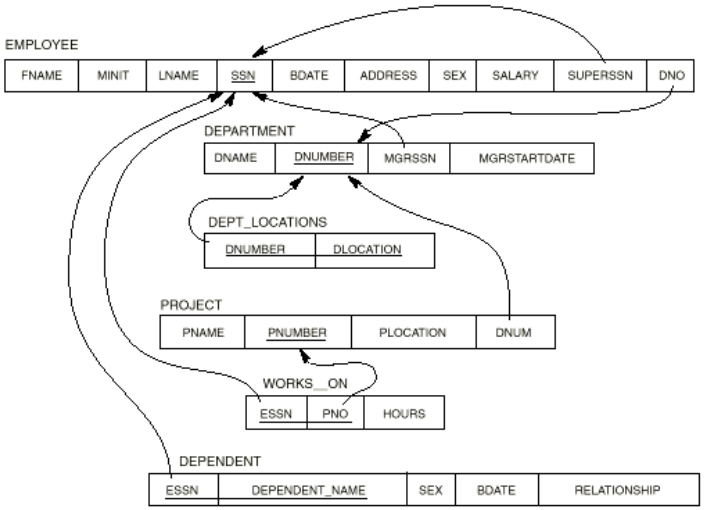
Restrições de Integridade Referencial

- Sejam
  - R e S relações
  - A = {A1,A2,..An} um conjunto de atributos de R
  - B = {B1,B2,..Bm} um conjunto de atributos de S
- Se existe uma restrição de integridade referêncial de A para B,então:

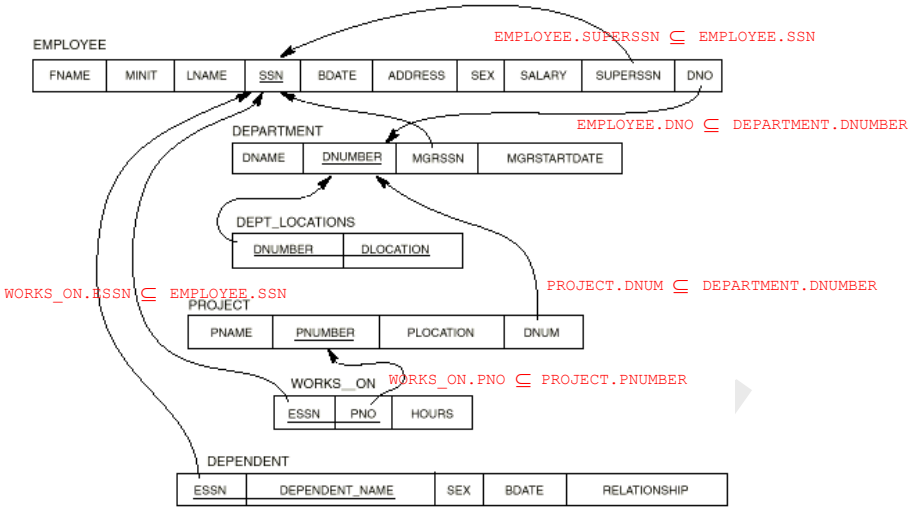
$$\pi_B(S) \subseteq \pi_A(R)$$

- A deve formar uma chave

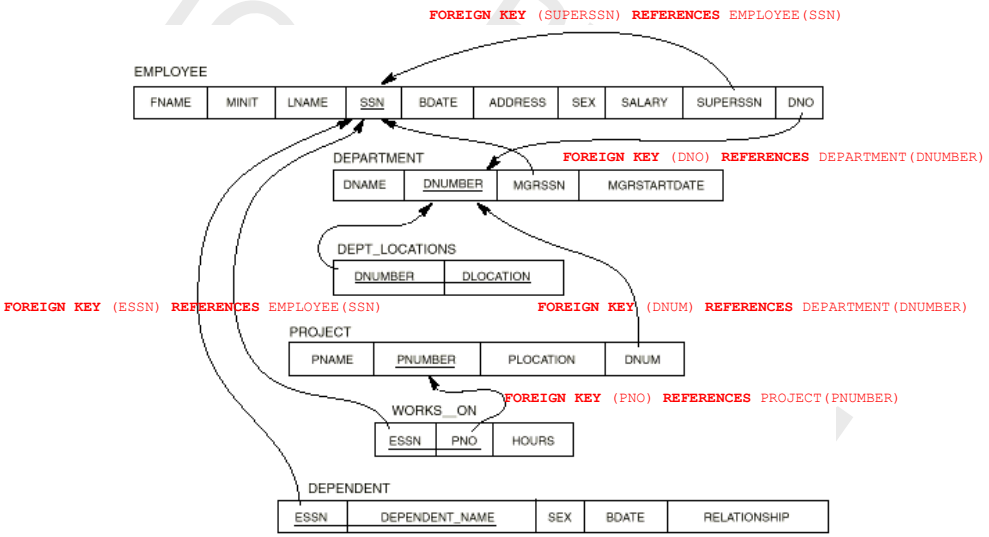
Restrições de Integridade Referencial



Restrições de Integridade Referências



Integridade Referencial em SQL



SQL Integridade Referencial

```
CREATE TABLE EMPLOYEE
    ( FNAME          VARCHAR(15)     NOT NULL ,
      MINIT          CHAR ,
      LNAME          VARCHAR(15)     NOT NULL ,
      SSN            CHAR(9)         NOT NULL ,
      BDATE          DATE
      ADDRESS        VARCHAR(30) ,
      SEX            CHAR ,
      SALARY         DECIMAL(10,2) ,
      SUPERSSN       CHAR(9) ,
      DNO            INT             NOT NULL ,
    PRIMARY KEY (SSN) ,
    FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN) ,
    FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNUMBER) ) ;

CREATE TABLE DEPARTMENT
    ( DNAME          VARCHAR(15)     NOT NULL ,
      DNUMBER        INT             NOT NULL ,
      MGRSSN         CHAR(9)         NOT NULL ,
      MGRSTARTDATE   DATE ,
    PRIMARY KEY (DNUMBER) ,
    UNIQUE (DNAME) ,
    FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN) ) ;

CREATE TABLE PROJECT
    ( PNAME          VARCHAR(15)     NOT NULL ,
      PNUMBER        INT             NOT NULL ,
      PLOCATION      VARCHAR(15) ,
      DNUM           INT             NOT NULL ,
    PRIMARY KEY (PNUMBER) ,
    UNIQUE (PNAME) ,
    FOREIGN KEY (DNUM) REFERENCES DEPARTMENT(DNUMBER) ) ;

CREATE TABLE WORKS_ON
    ( ESSN           CHAR(9)         NOT NULL ,
      PNO            INT             NOT NULL ,
      HOURS          DECIMAL(3,1)    NOT NULL ,
    PRIMARY KEY (ESSN, PNO) ,
    FOREIGN KEY (ESSN) REFERENCES EMPLOYEE(SSN) ,
    FOREIGN KEY (PNO) REFERENCES PROJECT(PNUMBER) ) ;
```

Expressão Chave Estrangeira

- Palavra-chave REFERENCES:
  - Após um atributo (para um atributo chave).
  - Como um elemento do esquema:

  FOREIGN KEY (<lista de atributos>)
  REFERENCES <relação> (<atributos>)

- Atributos referenciados devem ser declarados como PRIMARY KEY ou UNIQUE.

Exemplo: Com atributo

```
CREATE TABLE Cervejas (
  nome     CHAR(20) PRIMARY KEY,
  fabr     CHAR(20) );

CREATE TABLE Vendas (
  bar      CHAR(20),
  cerveja CHAR(20) REFERENCES Cervejas(nome),
  preco    REAL );
```

Exemplo: Como elemento do esquema

```
CREATE TABLE Cervejas(
  nome      CHAR(20) PRIMARY KEY,
  fabr      CHAR(20));

CREATE TABLE Vendas (
  bar       CHAR(20),
  cerveja   CHAR(20),
  preco     REAL,
 FOREIGN KEY(cerveja)REFERENCES Cervejas(nome));
```

Garantindo Restrição de Chave Estrangeira

- Se existe uma restrição de chave estrangeira da relação R para relação S, dois tipos de violações são possíveis:
  - Em uma inserção ou atualização em R introduzir valores não encontrados em S
  - Uma remoção ou atualização na relação S deixa algumas tuplas de R como “pendentes.”

Ações

- Suponha que *R* = Vendas, *S* = Cervejas.
- Uma inserção ou atualização na relação Vendas que insere uma cerveja que não existe em Cervejas, deve ser rejeitada.
- Uma remoção ou atualização em Cervejas que remove uma cerveja encontrada em alguma tupla de Vendas pode ser controlada de três maneiras

Ações

- Default : Rejeita a modificação.
- Cascade : Faz as mesmas alterações na relação Vendas.
  - Remove cerveja: remove tupla em Vendas.
  - Atualiza cerveja: altera o valor em Vendas.
- SET NULL: altera cerveja para NULL.

Exemplo: Cascade

- Remove a tupla que contem Bud da relação Cervejas:
  - Remove todas as tuplas de Vendas que tem cerveja = 'Bud'.
- Atualiza a tupla que contem Bud de 'Bud' para 'Budweiser':
  - Altera todas as tuplas de Vendas com cerveja = 'Bud' para cerveja = 'Budweiser'.

Exemplo: Set NULL

- Remove a tupla Bud de Cervejas:
  - Altera todas as tuplas de Vendas que tem cerveja = 'Bud' para cerveja = NULL.
- Atualiza a tupla Bud tuple de 'Bud' para 'Budweiser':
  - Mesma atualização que acontece na remoção.

Escolhendo uma política

- Quando declaramos uma chave estrangeira, podemos escolher uma política SET NULL ou CASCADE independentemente para remoção e atualizações.

ON [UPDATE, DELETE][SET NULL CASCADE]

- Caso contrário, o *default* (rejeita) é usado.

Exemplo: Configurando Política

```
CREATE TABLE Vendas (
  bar       CHAR(20),
  cerveja   CHAR(20),
  preco     REAL,
  FOREIGN KEY(cerveja)
     REFERENCES Cervejas(nome)
     ON DELETE SET NULL
     ON UPDATE CASCADE
);
```

Restrições e Gatilhos

- Uma restrição (constraint) é uma regra de consistencia entre os elementos de dados que o SGBD deve garantir.
  - Exemplo: Restrição de Chave.
- Gatilho (trigers) são procedimentos de verificação executados quando ocorre uma condição específica,
  - Exemplo: inserção de uma tupla.
  - Mais fácil implementar que restrições complexas.

Restrições Baseadas em Atributos

- Restrições sobre o valor de um atributo específico.
- Adicionar CHECK(<condição>) na declaração de um atributo.
- A condição pode usar o nome do atributo, porém qualquer outra relação ou nome de atributo deve estar na subconsulta.

Exemplo

```
CREATE TABLE Vendas (
  bar         CHAR(20),
  cerveja     CHAR(20)
   CHECK (cerveja IN
           (SELECT nome FROM Cervejas)),
  preco  REAL CHECK (preco <= 5.00 )
);
```

Tempo da Verificação

- Verificações baseadas em atributo são realizados somente quando um valor para este atributo é inserido ou atualizado.
  - Exemplo: `CHECK (preco <= 5.00)` checa cada novo preço e rejeita a modificação (para esta tupla) se o preço é maior que $5.
  - Exemplo: `CHECK (cerveja IN (SELECT nome FROM Cervejas))` não checa se uma cerveja é removida da relação Cervejas (diferente de chave estrangeira).

Restrições baseadas em Tuplas

- CHECK (<condition>) pode ser adicionado como elemento do esquema.
- A condição pode referenciar qualquer atributo da relação.
  - Mas outros atributos ou relações requerem uma subconsulta.
- Verifica somente inserção ou atualização.

Exemplo:Ver. baseada em Tupla

- Somente o Bar do Zeca pode vender cerveja por mais que $5:

```
CREATE TABLE Vendas (
    bar         CHAR(20),
    cerveja     CHAR(20),
    preco       REAL,
    CHECK (bar="Zeca's Bar" OR
           preco <= 5.00));
```

Asserções

- São elementos do esquema do banco de dados, como relações ou visões (views)
- Definido por:

  CREATE ASSERTION <nome>

  CHECK (<condição>);
- <condição> pode fazer referência para qualquer relação ou atributo no esquema do banco de dados.

Exemplo: Asserção

- Em Vendas(bar, cerveja, preco), nenhum bar pode cobrar em média mais que $5.

```
CREATE ASSERTION VerExploradores
  CHECK (
  NOT EXISTS (
  SELECT bar,AVG(preco) FROM Vendas
  GROUP BY bar
  HAVING 5.00 < AVG(preco)
  ));
```

Bares com um preço Médio acima de $5

Exemplo: Asserções

- Na relação Clientes(nome, ender, phone) e Bares(nome, ender, cnpj), não pode ter mais bares que consumidores.

```
CREATE ASSERTION VerBar CHECK (
  (SELECT COUNT(*) FROM Bares) <=
  (SELECT COUNT(*) FROM Clientes)
);
```

Sincronismo

- Em princípio, deve-se verificar cada asserção antes de cada modificação para qualquer relação do banco de dados.
- Um sistema inteligente pode observar que somente certas alterações poderia violar uma dada asserção.
  - Exemplo:Nenhuma alteração na relação Cervejas pode afetar VerBar.

Gatilhos: Motivação

- *Asserções* são poderosas, mas o SGBD frequentemente não pode garantir quando elas serão checadas.
- Verificações baseados em Atributo e em tupla são verificados em tempos conhecidos, mas não são poderosos.
- Gatilhos permitem que o usuário decida quando checar para qualquer condição.

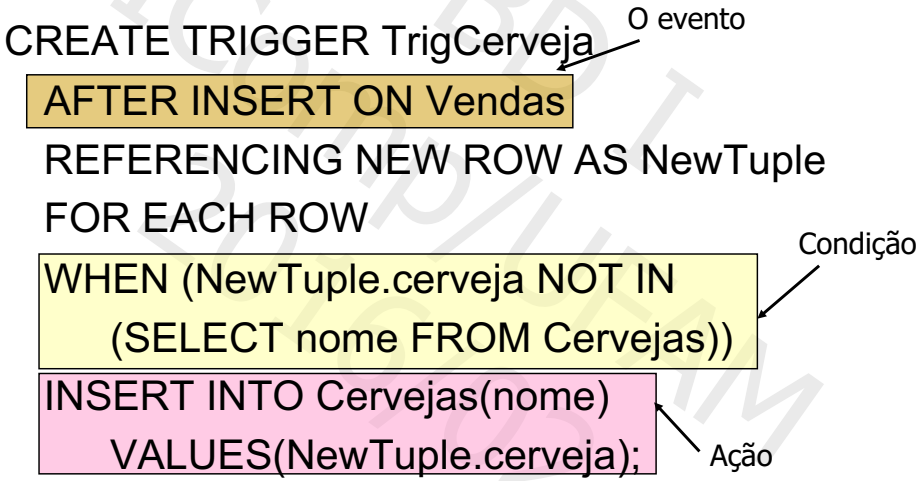
Regras *Evento-Condição-Ação*

- Outro nome para “gatilho” é regra *ECA,* ou regra *evento-condição-ação*
- *Evento*:
  - tipo de modificação do banco de dados, e.x., “insert on Vendas.”
- *Condição*:
  - Qualquer valor boleano em SQL.
- *Ação*:
  - Qualquer sentença SQL .

Exemplo:

- Em vez de usar restrição com chave estrangeira e rejeitar inserções em Vendas(bar, cerveja, preco) como cervejas desconhecidas, um gatilho pode adicionar essa cerveja em Cervejas, com um fabricante NULL.

Exemplo: Definição de Gatilho



Opções: Criando Gatilho

- CREATE TRIGGER <nome>
- CREATE OR REPLACE TRIGGER <nome>
  - Útil se existe um gatilho com esse nome, se deseja modificar o gatilho.

Opções: O Evento

- INSERT pode ser DELETE ou UPDATE.
  - Um UPDATE pode ser UPDATE . . . ON um atributo específico.

Opções: FOR EACH ROW

- Gatilhos a nível de Sentença :
  - Executa uma vez para a sentença SQL, sem considerar quantas tuplas foram modificadas
- Gatilhos a nível de Linha:
  - Executa uma vez para cada tupla modificada.
- FOR EACH ROW
  - usará “nível-linha”, senão “nível-sentença”.

Opções: REFERENCING

- INSERT implica em uma nova tupla (para nível de linha) ou em uma nova tabela (para nível de sentença).
  - A “tabela” é um conjunto de tuplas inseridas.
- DELETE implica em uma tupla antiga ou tabela.
- UPDATE implica ambos.
- [NEW | OLD][TUPLE | TABLE] AS <nome>

Opções: A Condição

- Qualquer condição de valor boleano.
- Avaliado no BD como seria antes ou depois do evento para o Gatilho, dependendo, se BEFORE ou AFTER é usado.
  - Mas sempre antes das alterações terem efeitos.
- A nova/antiga tupla/tabela pode ser acessada através dos nomes na cláusula REFERENCING.

Opções: A Ação

- Pode ser mais que uma sentença SQL na ação.
  - Formado por BEGIN . . . END se existe mais que uma.
- Mas consultas não fazem sentido em uma ação, então nós estamos realmente limitando para modificações.

Outro: Exemplo

- Usando Vendas(bar, cerveja, preco) e uma relação unária com Explorador(bar), mantenha uma lista de bares que tenham aumento no preço da cerveja maior que $1.

Gatilhos