SEGURANÇA E INTEGRIDADE

Prof. Heloise Manica

Introdução

- Os dados guardados em bancos de dados precisam estar protegidos de:
  - acessos não autorizados;
  - destruição ou alteração intencional;
  - inclusão acidental de inconsistências.
- O mal uso dos dados pode ser catalogado como:
  - intencional (maldoso) ou
  - acidental.

Violação da Segurança e Integridade

- A perda acidental da consistência dos dados pode resultar de:
  - Quedas durante o processamento da transação;
  - Anomalias motivadas por acesso simultâneo ao banco de dados;
  - Anomalias motivadas pela distribuição de dados em vários computadores;
  - Erro lógico que viola a suposição de que as transações devam preservar as restrições de consistência do banco de dados.

Violação da Segurança e Integridade

- É mais fácil a proteção contra perda acidental que acesso intencional.
- Formas de acesso intencional:
  - Leitura não autorizada de dados (furto de informação);
  - Modificação não autorizada de dados;
  - Destruição não autorizada de dados.
- Proteção absoluta não é possível
  - porém, mecanismos de proteção podem tornar o custo de acesso tão alto que iniba praticamente todas as tentativas de acesso não autorizado.

Violação da Segurança e Integridade

- O termo *segurança de banco de dados* normalmente refere-se à segurança contra acesso intencional, enquanto *integridade* refere-se a evitar a perda acidental de consistência.

Violação da Segurança e Integridade

- Medidas de segurança – Níveis:
  - Físico: local seguro;
  - Humano: reduzir a chance de um usuário autorizado dar acesso a um estranho, em troca de suborno ou outro favor;
  - Sistema operacional: devido a acesso remoto via terminais ou redes, segurança em nível de software;
  - Sistema de banco de dados: é responsabilidade do SBD assegurar que restrições relacionadas em nível de acesso do usuário não sejam violadas.

Violação da Segurança e Integridade

- Em nível de banco de dados, a segurança é responsabilidade do DBA
  - O DBA concede privilégios a usuários que necessitem usar o sistema e classifica os usuários e os dados de acordo com a política da organização
  - Os comandos de privilégio do DBA incluem comandos para conceder e revogar privilégios a contas individuais, a usuários ou a grupos de usuários.
  - O administrador do BD possui a forma suprema de autoridade. Ele pode autorizar novos usuários, reestruturar o BD, etc.

Autorização e Visões

- Embora um usuário não tenha acesso direto a uma relação, ele pode ter acesso a parte da mesma relação através de uma visão.
- Uma visão pode ocultar dados que o usuário não precisa ver.
- Sistema de banco de dados relacionais oferecem segurança em dois níveis:
  - Relação: pode permitir ou negar o acesso direto do usuário a uma relação;
  - Visão: pode permitir ou negar acesso a dados que apareçam numa visão.

Autorização e Visões

- Exemplo:
  - O funcionário precisa saber os nomes dos clientes de cada agência do banco;
  - Ele não tem autorização para ver informação relativa a empréstimos ou contas específicas que o cliente possa ter.
  - Assim, deve-se negar o acesso direto do funcionário às relações de *empréstimos* e *depósitos*.

Autorização e Visões

- Exemplo (cont...):
  - Relações:
    - Empréstimo (#agencia, #emprestimo, cliente, valor);
    - Depósito (#agencia, #conta, cliente, saldo).

Create view todos_clientes **as**

(**Select** #agencia, cliente **from** Depósito)

Union

(**Select** #agencia, cliente **from** Empréstimo)

  - Assim, o funcionário executa a consulta:

Select * **from** todos_clientes

Autorização

- Formas de autorização sobre partes do BD:
  - Autorização de leitura;
  - Autorização de entrada:
  - Autorização de atualização;
  - Autorização de eliminação;

Autorização

- Formas de autorização para modificar o esquema do BD:
  - Autorização de índice (permite criação e remoção de índices);
  - Autorização de recurso (permite a criação de novas relações);
  - Autorização de alteração (permite criação ou remoção de atributos em uma relação);
  - Autorização de redução (permite a remoção de relações).

Autorização

- A autorização é concedida usando a seguinte declaração SQL:

  grant <lista de privilégios> **on** <nome da relação ou da visão> **to** <lista de usuário>

  Ex.:

  grant update **on** empregado to U_1, U_2, U_3

Autorização

- Exemplos:
- **grant** insert, delete **on** *empregado*, *depto* **to** U_1
- **grant** select **on** *empregado*, *depto* **to** U_2 **with grant option**
- **grant** select **on** *empregado* **to** U_5
- **revoke** select **on** *empregado* **from** U_2
- **grant** update **on** *empregado* (*salario*) **to** U_1

Autorização

- A autorização é revogada usando a seguinte declaração SQL:

  revoke <lista de privilégios> **on** <nome da relação ou da visão> **from** <lista de usuário>

Ex.:

revoke update **on** *empregado* from U_1, U_2, U_3

Autorização

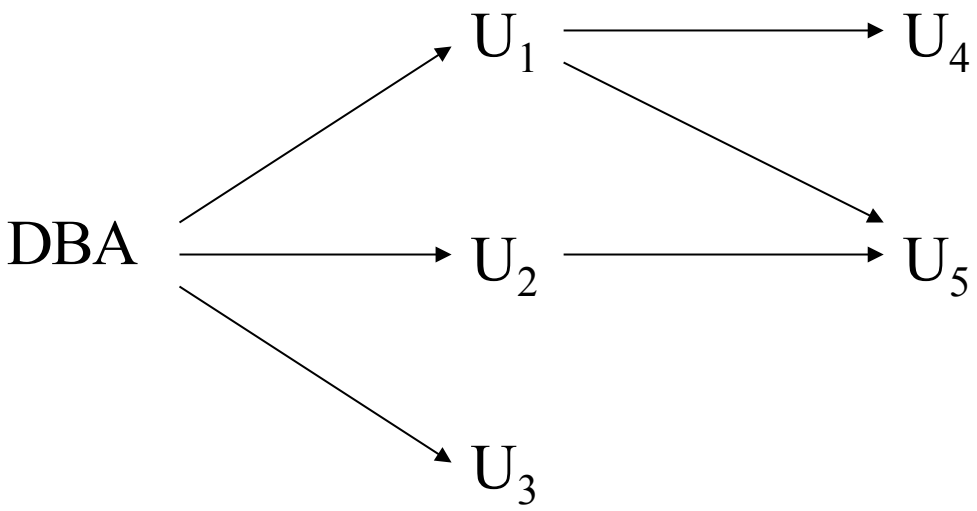
- Tipos de privilégios em SQL:
  - SELECT: dá o privilégio de recuperar tuplas de uma relação;
  - MODIFY: dá o privilégio para modificar tuplas de uma relação;
  - REFERENCES: dá a capacidade para referenciar uma relação quando especificando restrições de integridade.

Autorização

- Poderá ser permitido a um usuário, que recebeu alguma forma de autorização, transferir esta autorização a outros usuários.
- A transferência de autorização de um usuário a outro pode ser representada através de um *grafo de autorizações*
  - os nós são os usuários;
  - uma aresta (U_i, U_j) é incluída no grafo caso o usuário U_i conceda autorização de atualização sobre o *empregado* a U_j.

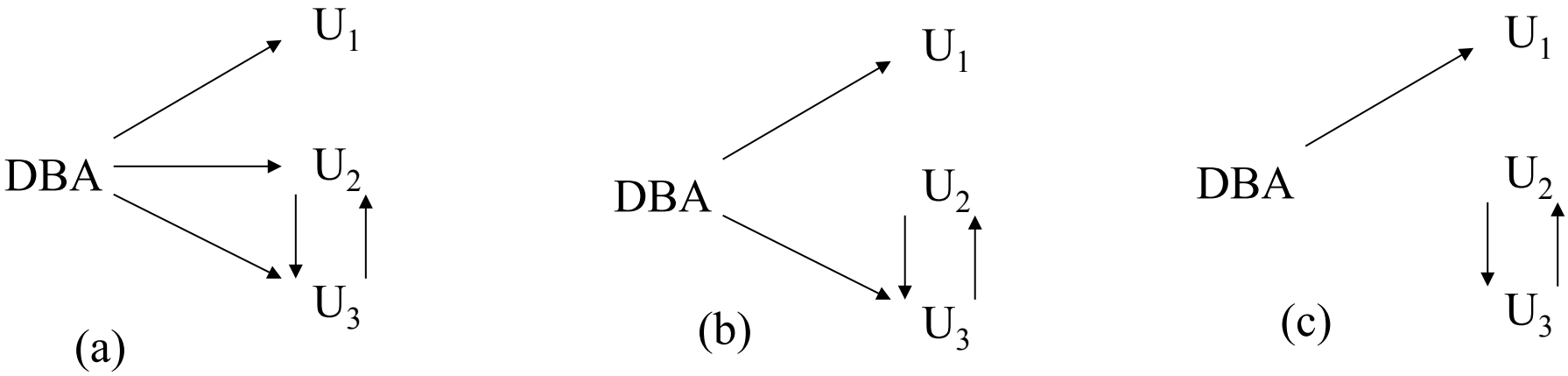
Autorização

- Exemplo de Grafo de Autorizações:



Autorização

- Exemplo de Grafo de Autorizações:



(a) Usuário mal intencionado pode tentar anular as regras de revogação concedendo autorizações entre si

(b) Se o DBA revogar autorização para U2, ele pode manter a autorização por meio de U3

(c) Se a autorização para U3 for revogada em seguida, ele pode manter a autorização por meio de U2

→ Por isso, exige-se que toda aresta, de um grafo de autorização, seja parte de algum caminho que tenha como origem o administrador do BD.

Autorização

- Para evitar que um par de usuários tente, intecionalmente, quebrar as regras relativas à revogação de autorização, mediante a concessão mútua de autorização, como mostram as figuras (a), (b), e (c), exige-se que toda aresta, de um grafo de autorização, seja parte de algum caminho que tenha como origem o administrador do BD.

Controle de Acesso

- Os comandos **grant** e **revoke** são do tipo "tudo ou nada" (*discretionary access control*)
- Algumas aplicações precisam de um esquema um pouco mais elaborado (governo, comando militar, aplicações inteligentes, indústrias, etc.)

Controle de Acesso

- Segurança em vários níveis: classificação de dados e usuários em níveis de segurança (*mandatory access control*)
- Níveis de segurança:
  - TS: *top secret*
  - S: *secret*
  - C: *confidential*
  - U: *unclassified*

TS>S>C>U

Controle de Acesso

- Modelo mais usado para segurança multinível

  - *Bell-LaPadula*:

    – sujeito (usuário, conta, programa)

      - **class (S)**

    – objeto (relação, tupla, coluna, visão, operação)

      - **class (O)**

Controle de Acesso

- Restrições:
  - um sujeito S não tem acesso de leitura a um objeto O a menos que class (S) ≥ class (O);
  - um sujeito S não pode gravar um objeto O a menos que class (S) ≤ class (O).
- A incorporação de segurança multinível em SGBD pode ser em valores de atributos ou em tuplas como objetos de dados.

Controle de Acesso

Níveis de segurança:
TS: *top secret*
S: *secret*
C: *confidential*
U: *unclassified*

- Exemplo:

  $R(A_1, C_1, A_2, C_2, \ldots, A_n, C_n, TC)$

  A = atributo
  C = classificação do atributo
  TC = classificação da tupla

  – O valor de TC deve ser o mais alto de todos os valores de classificação dos atributos na tupla.

Controle de Acesso

- Uma relação com multiníveis de segurança pode apresentar dados ≠ para sujeitos com ≠ níveis de classificação
  - filtragem: é possível armazenar uma única tupla na relação a um nível de classificação mais alto e produzir as tuplas correspondentes a um nível mais baixo de classificação.
  - Polinstanciação: várias tuplas podem ter o mesmo valor de chave, mas valores de atributos ≠ para usuários em ≠ níveis de classificação.

Controle de Acesso

Empregado (Nome, Salário, Níveltrabalho, TC)

a) Tuplas originais:

Maria C, 4000 S, Bom C, S

João U, 3000 C, Médio S, S

b) Filtragem para usuários classe C

Maria C, nulo C, Bom C, C

João U, 3000 C, nulo C, C

c) Filtragem para usuários classe U

João U, nulo U, nulo U, U

Níveis de segurança:
TS: *top secret*
S: *secret*
C: *confidential*
U: *unclassified*

Controle de Acesso

Níveis de segurança:
TS: *top secret*
S: *secret*
C: *confidential*
U: *unclassified*

d) Polinstanciação: suponha que um usuário com classe de segurança C tente alterar o valor do nível de trabalho de João para "Excelente", isto corresponde a:

update Empregado

set NivelTrabalho = "Excelente"

where Nome = "João"

Isto seria permitido, uma vez que os usuários de classe C receberam privilégio de atualização dessa relação, o sistema não teria como rejeitar.

Controle de Acesso

- O usuário poderia acreditar que algum valor não nulo existe para NívelTrabalho de João, considerando que o valor nulo é que aparece.
- Não deveria ser permitido ao usuário sobrepor o valor existente de NívelTrabalho no nível de classificação mais alto.

Controle de Acesso

- A solução é criar uma polinstanciação da tupla João ao nível de classificação C, como a seguir:

Maria	C,	4000	S,	Bom	C, S
João	U,	3000	C,	Médio	S, S
João	U,	3000	C,	Excelente	C, C

- As operações de atualização do modelo relacional (*insert*, *delete*, *update*) deveriam ser modificadas para manipular esta e outras situações similares.

Restrições de Integridade

- Integridade refere-se a precisão ou correção de dados em um banco de dados;
- Restrição refere-se a impor uma condição para qualquer atualização.
- As restrições de integridade protegem contra danos acidentais ao banco de dados.
- O objetivo é fornecer o meio de assegurar que as mudanças feitas no BD, por usuários autorizados, não levem à perda de consistência dos dados.

Restrições de Integridade

- Integridade referencial:
  - Assegura-se que o um determinado valor apresentado em uma relação (tabela) de acordo com o seu conjunto de atributos apareça em outro conjunto de atributos de outra relação.

- Exemplo:

  Sendo **"*Informática*"** um nome de departamento que aparece na tupla (linha) da relação (tabela) Departamento, então deve existir uma tupla **"*Informática*", ** na relação Empregado.

Restrições de Integridade

- Restrições de Domínio
  - Forma mais elementar em um Restrição de Integridade;
  - *Verificam* os valores inseridos no BD
  - Testam (***efetuam***) consultas para assegurar que as comparações façam sentido.
  - É realizada através da clausula **CHECK:**

Exemplo:

create domain hora_salario **double(5,2)**

constraint valor_teste **check**(valor >=4.00)

Restrições de Integridade

- O teste das restrições de domínio é análogo a teste durante a execução, no caso de uma linguagem de programação

- Exemplos:
  - Nenhum salário deve ser inferior a R$ 1000:
    - `check (salario >= 1000)`
  - O sexo do funcionário deve ter um dos valores M ou F:
    - `check(sex in ('M', 'F'))`
  - Nenhum salário deve ser inferior ao mínimo dos salários:
    - ```
      check (salario >= (select
        (min(salario) from Funcionarios))
      ```
      ```

Exemplos

create domain numero_conta **char (10)**
constraint teste_nulo_nconta **check**(**value not null**)

create domain tipo_conta **char (10)**
constraint teste_tipo_conta **check**(**value in (“corrente”,“Poupança”))**

```
mysql> CREATE TABLE pessoa3 (
-> idade INT(3) CHECK(idade>0 && idade<120),
-> sexo CHAR(1) CHECK(sexo IN ('M', 'F'))
);
```

Restrições de Integridade

- Uma ***asserção*** é um predicado expressando uma condição que queremos que o Banco de Dados sempre satisfaça (é verificada sempre que qualquer tabela é modificada)
- Restrições de Domínios e Regras de Integridades referencial são formas especiais de asserções.
- Exemplo:

create assertion <nome-asserção> **check** <predicado>

- Este teste pode introduzir uma quantidade significativa de sobrecarga; assim as asserções devem ser usadas com grande cuidado.

Restrições de Integridade
Asserção

- Exemplo:
  - O total de empréstimo de um cliente deve ser sempre inferior a R$ 5.000,00.

```
CREATE ASSERTION RestriçãoEmprestimo
    CHECK (5.000 >= ALL (SELECT SUM(valor) FROM Emprestimo
    GROUP BY numero_conta))
```

Restrições de Integridade

- Gatilho:
  - esquema alternativo para preservar a integridade;
  - declaração executada automaticamente pelo sistema, como efeito de uma modificação do BD;
  - deve-se especificar as condições sob as quais o gatilho será executado;
  - deve-se especificar as ações a serem tomadas quando o gatilho for executado.

Restrições de Integridade

- Exemplo:
  - Supondo que no lugar de permitir saldos negativos de conta, o banco solucione as contas descobertas deixando o saldo da conta em zero e criando um empréstimo equivalente ao valor descoberto;
  - A condição para executar o gatilho é uma atualização da relação *depósito*, que resulte num valor negativo de *saldo*;
  - Seja *t* a tupla com valor negativo, as ações a serem tomadas são as seguintes:
    - Inserir uma nova tupla *s* na relação *empréstimo*:
    - Colocar *t*[*saldo*] no valor zero.

Restrições de Integridade

define trigger saque-descoberto
on update of conta T
(if new T.saldo < 0 **then (insert into** emprestimo **values**
(T.nome-agencia,T.numero-conta, - **new** T.saldo)

insert into devedor **(select** nome-cliente, numero-conta
from depositante
where T.numero-conta = depositante.numero-conta)
update conta S
set S.saldo =0
where S.numero-conta =T.numero-conta))

Codificação (Criptografia)

- Utilizada quando os métodos adotados para a autorização não sejam suficientes para a proteção de dados altamente sensíveis.
- Não é possível ler dados codificados, a menos que o leitor conheça como decifrá-los (decodificá-los).
- Existe um vasto número de técnicas para a codificação de dados.

Codificação

- Técnica simples: substituição de cada caracter do alfabeto pelo próximo caracter.
  - Maringá → Nbsjohb
- Propriedades de boas técnicas:
  - É relativamente simples, para um usuário autorizado, codificar e decodificar dados;
  - O esquema de codificação não depende do segredo do algoritmo, mas de parâmetro do algoritmo, chamado chave de codificação;
  - É difícil para um intruso determinar a chave de codificação.

Codificação

- Norma de Codificação de Dados (*Data Encryption Standard* - DES)
  - realiza, ao mesmo tempo, a substituição de caracter e sua reordenação com base numa *chave de codificação*;
  - ponto débil: transferência da chave de codificação.

Codificação

- Codificação com chave pública: baseado em duas chaves, uma pública e outra privada.
  - Esquema de codificação aberto ao público para codificação com chave pública, baseia-se em:
    - existe um algoritmo eficiente para testar se um número é primo ou não;
    - não existe algoritmo eficiente conhecido para achar os fatores primos de um número.

Codificação

- os dados são tratados como uma coleção de inteiros;
- cria-se uma chave pública calculando o produto de dois grandes números primos P_1 e $P_{2;}$
- a chave privada consiste no par (P_1 e P_2);
- o algoritmo de decodificação não pode ser acionado com sucesso se somente é conhecido o produto P_1P_2.

Banco de Dados Estatísticos

- Bancos de dados estatísticos são usados para produzir estatísticas sobre uma população variada;
- Podem ser realizados somente estudos estatísticos através das funções:
  - COUNT, SUM, MIN, MAX, AVERAGE e STANDARD DEVIATION.

Banco de Dados Estatísticos

- Um ponto fraco em BD estatísticos é constituído por casos raros
  - Exemplos:
    - O sistema poderia divulgar dados individuais quando fosse solicitado o total de saldos bancários de todos os clientes que moram numa cidade onde, na realidade, morasse somente um cliente do banco.
    - Sabendo que Maria mora em Maringá e que ela é PH.D, a seguinte consulta poderia ser feita:

      SELECT SUM(SALARIO) **FROM** CLIENTE
      WHERE (ÚLTIMO_NÍVEL='PH.D.' **AND**
      SEXO = 'F' **AND** CIDADE = 'MARINGÁ')

Banco de Dados Estatísticos

- Para o primeiro caso, uma solução é exigir que o sistema rejeite qualquer solicitação que abranja um número de indivíduos menor do que aquele predeterminado *n*.
- Para o segundo caso, uma solução seria proibir seqüências de consultas que fazem referência, repetidamente, a mesma população de tuplas, ou seja, exigir-se que não exista um par de solicitações com intersecção maior que *m*.

Banco de Dados Estatísticos

- Essas duas restrições não excluem a possibilidade de alguma solicitação extremamente inteligente que revele dados individuais. Porém, pode provar-se que serão necessárias pelo menos 1+(*n*-2)/*m* solicitações para que o usuário maldoso possa determinar os dados relativos a um indivíduo.

Banco de Dados Estatísticos

- Outro esquema de segurança é a poluição de dados
  - Implica na falsificação aleatória dos dados fornecidos como resposta a uma consulta;
  - deve ser feita de tal modo que o significado estatístico da resposta não seja destruído.