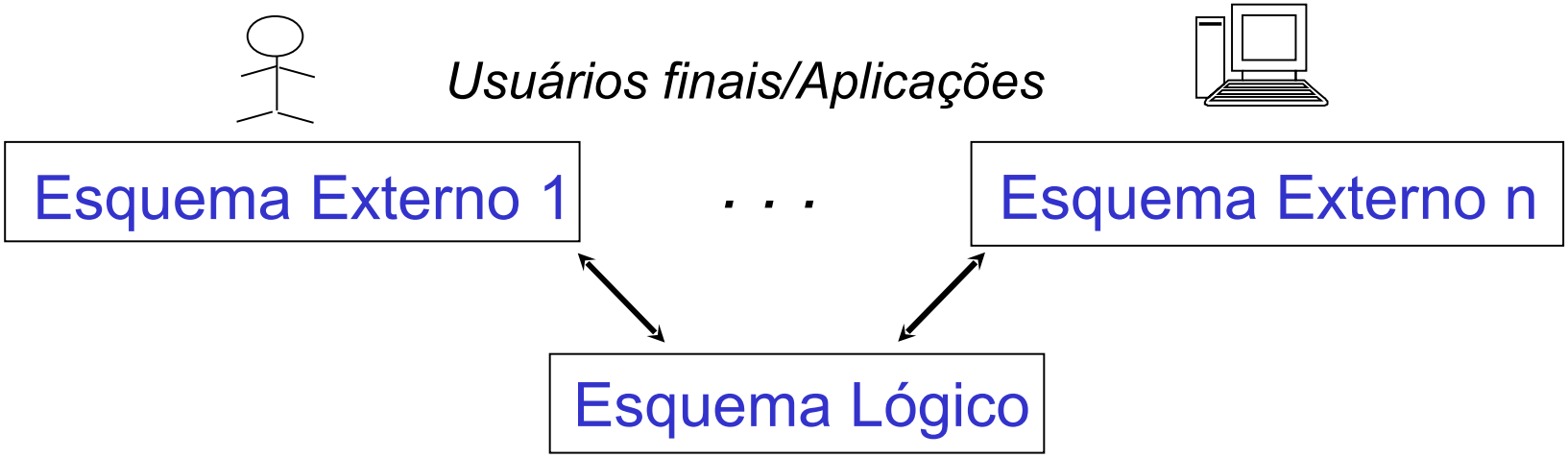
# Visões

- Visão
  - tabela derivada a partir das tabelas do BD
  - tabela virtual
    - isto é transparente para usuários e aplicações
      - visões são manipuladas como tabelas normais do BD
- Visões fazem parte dos esquemas externos da arquitetura de um SGBD



# Definição de Visões

- Exemplo
  - o setor de câncer do hospital lida apenas com dados de pacientes que têm esta doença
- Definição em SQL

```
create view PacCâncer
(código, paciente, idade)
  as
   select codp, nome, idade
   from Pacientes
   where problema = 'câncer';
```

# Definição de Visões

- Especificação da visão mantida no DD
  - a consulta que a define
- Definições recursivas são permitidas

```
create view PacCanJovens as
      select *
      from PacCâncer
      where idade < 21;
```

- Dependência Visão X Tabela/Visão Base

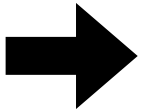
```
drop {table | view} nome [restrict | cascade]
```

(facilidade não encontrada em todos os SGBDs)

# Operações DML sobre Visões

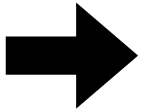
- São executadas na(s) tabela(s) base
- Exemplos

```
select *
from PacCâncer
where paciente like 'J%';
```



```
select codp, nome, idade
from Pacientes
where problema = 'câncer'
and nome like 'J%';
```

```
delete from PacCâncer
where idade > 90;
```



```
delete from Pacientes
where problema = 'câncer'
and idade > 90;
```

- Considerações
  - toda visão é passível de consulta
  - nem toda a visão é passível de atualização

# Atualização de Visões

- Situação 1

```
create view DadosMed as
    select codm, nome, CPF, especialidade
    from Médicos;

create view DadosPac as
    select nome, idade, cidade, doença
    from Pacientes;
```

⟶ I nas visões DadosMed e DadosPac?

# Atualização de Visões

- Situação 1

```
create view DadosMed as
    select codm, nome, CPF, especialidade
    from Médicos;

create view DadosPac as
    select nome, idade, cidade, doença
    from Pacientes;
```

⟶ I nas visões DadosMed e DadosPac?
  - chave primária sem valor!

- Conclusão 1
  - uma visão atualizável deve preservar as chaves candidatas dos dados da tabela base
    - mapeamento 1-1 entre tupla da visão e tupla da base

# Atualização de Visões

- Situação 2

```
create view ConsultasMedPac
(med, pac, nroConsultas) as
        select codm, codp, count(*) as total
        from Consultas
        group by codm,codp;
```

⟶ I / A na visão ConsultasMedPac?

# Atualização de Visões

- Situação 2

```
create view ConsultasMedPac
(med, pac, nroConsultas) as
        select codm, codp, count(*) as total
        from Consultas
        group by codm,codp;
```

⟶ I / A na visão ConsultasMedPac?

- atributo `total` não é atualizável!

- Conclusão 2
  - uma visão atualizável deve conter atributos que tenham correspondência direta com atributos da tabela base
    - mapeamento 1-1 entre atributo da visão e atributo da tabela base

# Atualização de Visões

- Situação 3
  - supor que a chave primária na tabela Consultas é (codm, codp, data, hora)

```
create view ConsultasMP
(codigoM,especialidade,codigoP, data) as
  select Médicos.codm, especialidade, codp, data
  from Médicos join Consultas
  on Médicos.codm = Consultas.codm;
```

# Atualização de Visões

- Situação 3

```
create view ConsultasMP
(codigoM,especialidade,codigoP, data) as
  select Médicos.codm, especialidade, codp, data
  from Médicos join Consultas
  on Médicos.codm = Consultas.codm;
```

⟶ intenção: o médico de código 13 transferiu as suas consultas do dia 29/10/10 para o médico de código 15

```
update ConsultasMP
set codigoM = 15
where códigoM = 13 and data = '10/29/2010';
```

# Atualização de Visões

- Situação 3

```
create view ConsultasMP
(codigoM,especialidade,codigoP, data) as
  select Médicos.codm, especialidade, codp, data
  from Médicos join Consultas
  on Médicos.codm = Consultas.codm;
```

⟶ intenção: o médico de código 13 transferiu as suas consultas do dia 29/10/10 para o médico de código 15

```
update ConsultasMP
set codigoM = 15
where códigoM = 13 and data = '10/29/2010';
```

⟶ Problemas: - nulos em atributos da chave primária
- inclusão em várias tabelas (?)

# Atualização de Visões

- Situação 3

```
create view ConsultasMP
(codigoM,especialidade,codigoP, data) as
   select Médicos.codm, especialidade, codp, data
   from Médicos join Consultas
   on Médicos.codm = Consultas.codm;
```

- Conclusão 3
  - uma visão atualizável deve ser derivada de apenas uma (1) tabela base
    - garantia de um mapeamento sempre 1-1 entre uma tupla da visão e uma tupla do BD

# Atualização de Visões

- Atributo definido no predicado da visão não é um atributo da visão

```
create view PacCâncer (código, paciente, idade)
as
    select codp, nome, idade
    from Pacientes
    where doença = 'câncer';
```

  - tuplas incluídas em PacCâncer não serão vistas por ela!
    - solução: *trigger* que define **doença = "câncer"**

# Atualização de Visões

- Atributo definido no predicado da visão é um atributo da visão

```
create view MédicosJovens as
   select codm, nome, especialidade, idade
   from Médicos
   where idade < 35;
   [ with check option; ]
```

  - cláusula *with check option*
    - controle de valores válidos para os atributos da visão que fazem parte do seu predicado
      - inclusão ou alteração em MédicosJovens
        - » apenas `idades < 35` serão permitidas!

# Uso de Visões

- Vantagens
  - independência lógica dos dados
    - cada usuário ou aplicação “enxerga” a porção do BD que deseja
  - acesso eficiente
    - plano de acesso otimizado!
  - segurança de acesso
    - cada usuário ou aplicação “enxerga” a porção do BD que tem permissão
- Problemática de visões atualizáveis
  - limita o uso de visões
  - tema de pesquisa na área de BD

Obs.: Postgres não suporta visões atualizáveis. Utilizar CREATE RULE para gerar uma atualização em tabelas a partir de uma atualização em uma visão

# Autorização de Acesso

- Objetivo
  - proteção contra acessos mal intencionados
- Subsistema de Autorização de Acesso (SAA)
  - controla quais dados um usuário/grupo de usuários pode ter acesso
  - controla quais operações um usuário/grupo de usuários pode realizar sobre estes dados
- Funções do subsistema de autorização
  - especificação de autorizações
  - verificação de autorizações

# Funções do Subsistema de AA

- Cadastro de usuários/grupos
  - *login* + *password*
- Especificação de autorizações
  - envolve três dimensões
    - agente (usuário ou grupo)
    - grânulo (BD, tabela, atributos, atributo, tuplas, ...)
    - operação (*select*, *update*, *create*, ...)
  - DBA
    - super usuário (pode tudo!)
      - alguns privilégios são exclusivos dele
        - » *recovery* do BD, configuração de parâmetros do SGBD, ...
    - concede/retira (revoga) privilégios de acesso
  - outros agentes
    - todos os privilégios de acesso aos grânulos (BDs e tabelas) que criou
    - concede/revoga privilégios para estes grânulos a outros agentes

# Classificação de AA

1. Baseadas no grânulo + operação
   - é ou não é válido para todos os usuários
     - permissões públicas ou secretas
1. Baseadas nas três dimensões
   - grânulo + operação + agente
   - utiliza matrizes de autorização de acesso
1. Baseadas em restrições
   - utiliza visões

# Matriz de AA

| | $G_1$ | $G_2$ | $G_3$ | . . . | $G_n$ |
|---|---|---|---|---|---|
| $A_1$ | *select* | *select, insert* | | | *create* |
| $A_2$ | | | *select, update* | | |
| $A_3$ | *select, delete, exec* | | | | *select, update, grant* |
| ... | | | | | |
| $A_m$ | | *create* | *select, insert* | | |

# Exemplo de Matriz AA: SQL Server

- Uma matriz por usuário/grupo
  - para cada BD

| | *select* | *update* | *insert* | *. . .* | *exec* |
|---|---|---|---|---|---|
| ambulatórios | X | | | | |
| médicos | X | | X | | |
| ... | | | | | |
| consultas | X | | | | X |

# Visões

- Maior flexibilidade na definição de grânulos
  - independentes ou dependentes de valor
- Exemplos

```
create view Med as
  select codm, nome, CPF, especialidade
  from Médicos
```

acesso apenas a alguns atributos **(indep. valor)**

```
create view MedOrtoped as
  select codm, nome, idade
  from Médicos
  where especialidade = 'ortopedia'
```

acesso apenas a alguns atributos e tuplas **(dep. valor)**

# Outras Considerações sobre AA

- Premissa básica
  - "quem não consulta não pode atualizar"
- Administrar corretamente permissões sobre tabelas e visões
  - responsabilidade do DBA
  - exemplo
    - não faz sentido uma mesma permissão sobre uma tabela base e uma visão derivada dela

# Autorização de Acesso em SQL

- Concessão de acesso

```
grant {all privileges | listaPrivilégios} on grânulo
   to {listaUsuários | public} [with grant option]
|
grant all to {listaUsuários | public}
```

- Exemplos

```
grant select, insert on Médicos to U1, U2
   with grant option
grant update on (idade, problema) Pacientes to U2
grant select on Consultas to public
grant all privileges on Ambulatórios to U1,U2, U4
grant exec on RemoveConsultasAntigas to U1
grant all to U5
```

# Autorização de Acesso em SQL

- Retirada de acesso

```
revoke[grant option for]{all privileges |
listaPrivilégios} on grânulo from listaUsuários[cascade]
|
revoke all from {listaUsuários | public}
```

- Exemplos

```
revoke all privileges on Ambulatórios from U1, U4
revoke delete on Pacientes from U3
revoke select on Médicos from U1 cascade
revoke all from U5
```