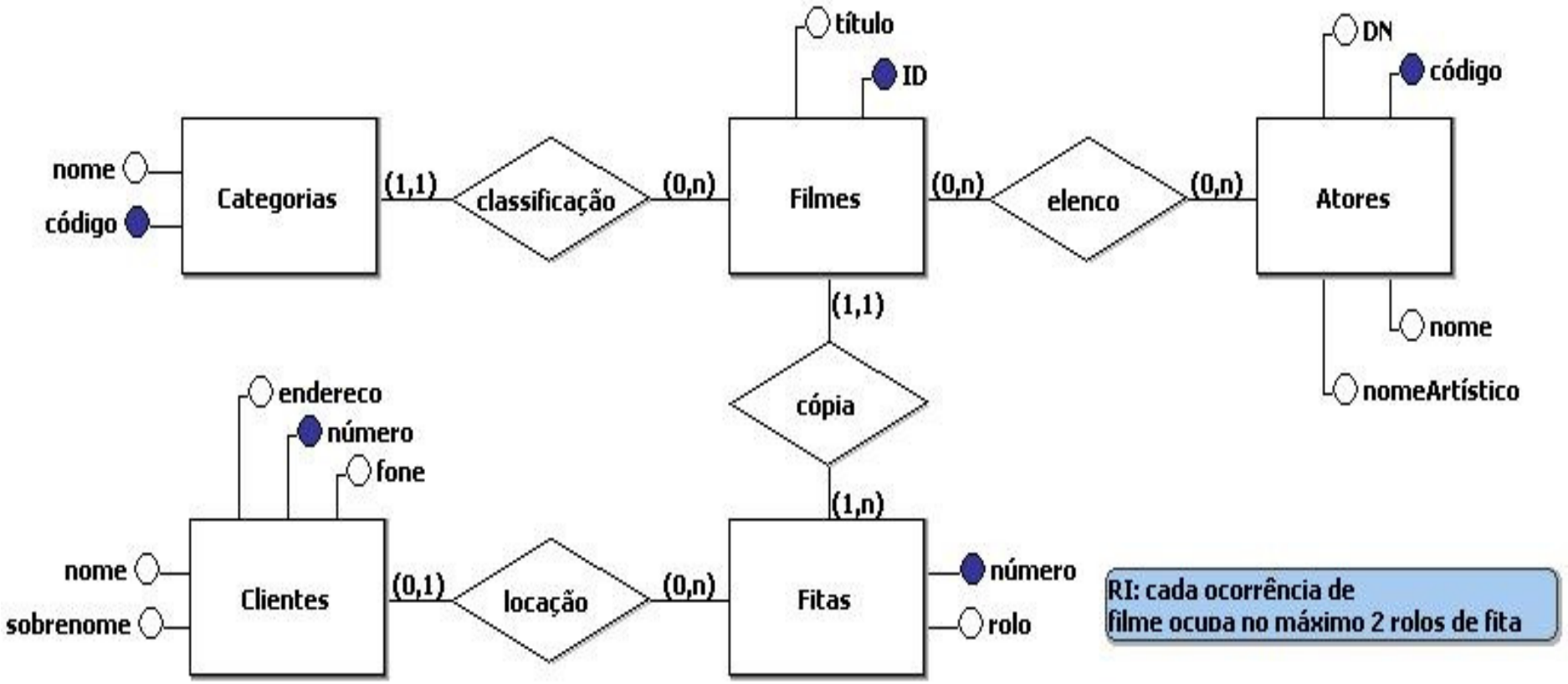
Dicas de Projeto Lógico Relacional

- O que deve ser especificado?
  - mapeamento do esquema conceitual
    - definição das tabelas e chaves
    - justificativas de mapeamento (se necessário)
  - restrições de integridade (RIs)
    - integridade referencial
    - restrições de domínio
      - valores e transições de valores permitidos
      - pode ocorrer checagem predicados e disparo em gatilhos
    - transações e procedimentos persistentes
      - podem ser definidos para encapsular conjuntos de operações sobre os dados
        - » em sintonia com o projeto da aplicação
  - visões
  - grupos de usuários e suas permissões

Dicas de Projeto Lógico Relacional

- Vantagens de uso de uma visão (*view*)
  - traduzida e otimizada "uma única vez" pelo SGBD
    - gera um plano ótimo de acesso a dados uma única vez
  - encapsula uma consulta complexa
    - relatório ou consulta freqüente ao BD
    - evita que a consulta seja formulada explicitamente pelo usuário toda vez que for necessária
      - pode ser formulada de várias formas, nem sempre ótima
      - requer tradução e otimização a cada formulação
- Vantagens do uso de um procedimento persistente (*stored procedure*)
  - reduz o código da aplicação
  - invocado por qualquer aplicação que acessa o BD
  - traduzido e otimizado para acesso a dados "uma única vez" pelo SGBD

Estudo de Caso (Parcial) – Vídeo Locadora VHS



Estudo de Caso – Vídeo Locadora

- Supor dois grupos de usuários
  - Funcionários
  - Clientes
- Especificação do Esquema Relacional

  Categorias (*código*, *nome*)

  Funcionários: I, E, A, C

Estudo de Caso – Vídeo Locadora

Filmes (*id*, *título*, $\overline{\textit{categoria}}$)

Funcionários: I, E, A, C

RIs:
a) *categoria* não nulo
b) *categoria* faz referência a Categorias
E: impedimento
A: cascata
c) I ou A com *categoria* inexistente:
impedimento
d) transação:
I de f $\in$ Filmes $\Rightarrow$ I de ft $\in$ Fitas :
ft.*filme* = f.*id* (pelo menos 1 tupla)

Estudo de Caso – Vídeo Locadora

Filmes (cont.)

visão FilmeCat(f.*id*, f.*título*, count(ft.*número*))
com f ∈ Filmes, ft ∈ Fitas;
ft.*filme* = f.*id*; e
count(ft.*número*) indica a quantidade de fitas do filme que estão disponíveis (não locadas)

Funcionários, Clientes: C

Estudo de Caso – Vídeo Locadora

Fitas (*número*, *rolo*, *filme*, *locadoPara*)

Funcionários: I, E, A, C

RIs:
a) *filme* não nulo
b) *filme* faz referência a Filmes
   E: impedimento
   A: cascata
c) I ou A com *filme* inexistente:
   impedimento
d) *locadoPara* faz referência a Clientes
   E: impedimento
   A: cascata

Estudo de Caso – Vídeo Locadora

Fitas (cont.)

RIs: e) I com *locadoPara* inexistente:
anulação
f) A com *locadoPara* inexistente:
impedimento
g) rolo ∈ {1, 2}

Justificativa para o mapeamento do relacionamento Locação:

- 1 fita pode ou não estar emprestada para no máximo 1 cliente
  - chave estrangeira em Fitas economiza espaço
- não existem atributos no relacionamento
  - apenas um atributo adicional na tabela Fitas para estabelecer o relacionamento

Estudo de Caso – Vídeo Locadora

Fitas (cont.)

procedimento Empréstimo(f Fitas, c Clientes)
  A de f: f.*locadoPara* ← c.*número*

procedimento Devolução(f Fitas, c Clientes)
  A de f: SE f.*locadoPara* = c.*número*
  ENTÃO f.*locadoPara* ← NULL

Estudo de Caso – Vídeo Locadora

Atores (*código, DN, nome, nomeArtístico*)

Funcionários: I, E, A, C

RI: *nome* ≠ *nomeArtístico*

Clientes (*número, fone, end, nome, sobrenome*)

Funcionários: I, E, A, C
Clientes: A, C do seu cadastro pessoal (*login* = *número*) para *fone*, *end*, *pNome* e *sNom*

Estudo de Caso – Vídeo Locadora

Elenco (*filme*, *ator*)

Funcionários: I, E, A

RIs:

a) *filme* faz referência a Filmes
   E: cascata
   A: cascata

b) I ou A com *filme* inexistente:
   impedimento

c) *ator* faz referência a Atores
   E: cascata
   A: cascata

d) I ou A com *ator* inexistente:
   impedimento

Estudo de Caso – Vídeo Locadora

Elenco (cont.)

visão FilmeAt(f.*título*, a.*nomeArtístico*)
com f ∈ Filmes, a ∈ Atores; e ∈ Elenco; e
e.*filme* = f.*id* e e.*ator* = a.*código*

Funcionários, Clientes: C

Estudo de Caso – Vídeo Locadora

- Exemplo de gatilho
  - modificação nos requisitos: quando um ator não participar do elenco de nenhum filme, ele não deve ser mais mantido no BD

gatilho RemoveAtor

Evento: após E de f $\in$ Filmes

Condição: dado a $\in$ Atores : SE $\nexists$ e $\in$ Elenco

(e.*ator* = a.*código*)

Ação: E de a

Estudo de Caso – Vídeo Locadora

- Exemplo de procedimento para fins de RI
  - modificação nos requisitos:
    - se um cliente não loca filmes há mais de 2 anos (730 dias), ele deve ser removido do BD
    - supor que existe um histórico de locações em uma tabela chamada Empréstimos(fita, cliente, data)

procedimento RemoveClientesLixo

$\forall$ c $\in$ Clientes :

dado e $\in$ Empréstimos : e.*data* = max(e.*data*)

para e.*cliente* = c.*número*

SE *DATASISTEMA* - e.data > 730

ENTÃO E de c

Exercício de Projeto Conceitual/Lógico

DOMÍNIO: MUSEU

Cada obra no museu possui um *código*, um *título* e um *ano*. Obras ou são pinturas ou são esculturas. No primeiro caso, é importante registrar o *estilo* (por exemplo, impressionista). No caso de esculturas, são importantes o *peso* e o *material* de que é feito (argila, ferro etc). Uma obra pode estar exposta em um único salão, em uma determinada *posição* neste salão. Um salão, que geralmente abriga várias obras, é identificado por um *número* e está em um *andar* do museu. Certos dados a respeito dos autores de cada obra também são relevantes: *código*, *nome* e *nacionalidade*. Uma obra é produzida por apenas um autor, porém, pode existir mais de uma obra de um mesmo autor no museu. No museu trabalham restauradores de obras, que possuem um *ID*, *CPF*, *nome*, *salário* e *especialidades* (argila, pintura a óleo etc). Um restaurador pode estar realizando a manutenção de várias obras. Uma obra, caso esteja em manutenção, está nas mãos de apenas um restaurador. Para cada manutenção deve-se registrar a *data de início* e a *data prevista de término* do trabalho, uma *descrição* do serviço a ser feito e um *custo* previsto para realizar a manutenção. Uma manutenção pode estar utilizando uma ou mais matérias-primas. Uma matéria-prima possui um *código*, um *nome* e uma *quantidade* em estoque. Uma matéria-prima pode estar sendo utilizada em várias manutenções, em uma certa *quantidade*.

Dicas de Projeto Físico Relacional

- Importância desta etapa
  - definição de estratégias de acesso para maximizar o desempenho
- Estratégias de acesso
  - devem ser constantemente revistas e ajustadas pelo DBA durante o tempo de vida do BD (*Tuning* do BD)
    - análise de processamento de transações e da organização/volume dos dados
- Dicas importantes
  - Implementação de consultas
  - Índices
  - Configurações do BD e de transações
  - Revisões do projeto lógico

Implementação de Consultas

- Definir expressões SQL com melhor desempenho possível
  - processadores de consultas de SGBDs podem ter capacidades limitadas de otimização
- Exemplos

```
SELECT f.título, a.nomeArt
FROM Filmes f, Elenco e, Atores a
WHERE e.filme = f.id e e.ator = a.código
```

otimização

```
SELECT f.título, a.nomeArt
FROM Filmes f JOIN Elenco e ON e.filme = f.id
     JOIN Atores a ON e.ator = a.código
```

Implementação de Consultas

- Definir condições SQL com melhor desempenho possível
  - filtragens AND são melhores que filtragens OR
  - filtragens NOT(predicado) devem ser evitadas

a) $\neg (p1 \vee p2) \equiv (\neg p1) \wedge (\neg p2)$

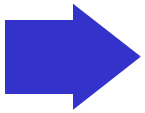
b) $\neg (p1 \wedge p2) \equiv (\neg p1) \vee (\neg p2)$

c) $p1 \vee (p2 \wedge p3) \equiv (p1 \vee p2) \wedge (p1 \vee p3)$

d) $p1 \wedge (p2 \vee p3) \equiv (p1 \wedge p2) \vee (p1 \wedge p3)$

  - exemplo (item a)

```
SELECT *
FROM Fitas ft JOIN Filmes f
ON ft.filme = f.numero
WHERE NOT
   (f.numero = 10 OR
   ft.locadoPara IS NULL)
```

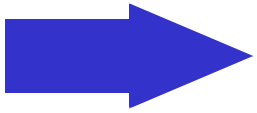


```
SELECT *
FROM Fitas ft JOIN Filmes f
ON ft.filme = f.numero
WHERE f.numero <> 10
AND
ft.locadoPara IS NOT NULL
```

Implementação de Consultas

- Definir predicados SQL com melhor desempenho possível
  - ordem dos predicados de uma condição muitas vezes é relevante
    - exemplo: utilize **`p1`** `OR p2` ao invés de `p2 OR p1` se o predicado **`p1`** é mais provável de ser verdadeiro
  - Valores constantes geram filtros mais seletivos
    - exemplo:

```
SELECT *
FROM ...
WHERE a.x = b.y
...
AND b.y = 100
```



```
SELECT *
FROM ...
WHERE a.x = 100
...
```

Índices

- Vantagens do uso de índices
  - mantém apontadores físicos
    - maior rapidez no acesso a dados
  - são estruturas de dados mais compactas
    - grande parte fica na memória principal
    - agilizam o processamento de predicados de igualdade por atributos indexados
      - teste feito em memória
- Quando se recomenda o uso de um índice
  - atributos envolvidos em predicados de seleção ou de junção em consultas freqüentes
  - atributos de tabelas com grande volume estimado de dados e passíveis de consultas com certa freqüência

Índices

- Quando se recomenda o uso de índices
  - atributos que são chaves alternativas
    - estrutura do índice é menor
      - » evita listas de apontadores auxiliares para acessar cada ocorrencia de valor, no caso de atributo não-chave
  - atributos envolvidos em transações “críticas”
    - exigem tempo de resposta muito pequeno
- Quando não se recomenda o uso de índices
  - atributos que sofrem muita atualização e pouca consulta
    - exigem atualização freqüente da estrutura de índices
- Índices compostos
  - geram estruturas mais complexas de gerenciar
  - recomendados somente quando dois ou mais atributos são testados sempre em conjunto
    - exemplo: consulta ao estoque de calçados de uma loja
      - verificação conjunta de *(tipo_calçado, numeração)*

Índices

- Índices em árvore
  - indicados para atributos não-chave e para predicados de desigualdade (<, >=, intervalo de valores, etc)
    - algoritmos baseados em buscas sobre sub-árvores que mantêm dados ordenados
- Índices *hash*
  - indicados para predicados de igualdade ou junções envolvendo atributos chave ou `UNIQUE`
    - acesso a poucos blocos de índice
  - indicados para tabelas dinâmicas
    - crescem e encolhem com freqüência
    - reorganização de estruturas *hash* é rápida

Configurações de Transações e BD

- *Buffer*
  - tamanho
    - geral (dados, índices, *logs*, ...), da *cache* de dados, ...
  - gerência
    - exemplo: *force / not force*
- Técnicas de processamento de consultas
  - exemplo: executar/não executar *sort* em junções
- *Timeout* da transação
- Número máximo de transações concorrentes
- Controle de serializabilidade
  - pode ser relaxada para transações só de leitura ou só de atualização
  - em SQL-2 ANSI: **`SET TRANSACTION`** ***`modoAcesso grauIsolamento`***
  - Exemplos:
    - `SET TRANSACTION WRITE ISOLATION LEVEL SERIALIZABLE ... COMMIT` (isolamento completo)
    - `SET TRANSACTION READ ISOLATION LEVEL READ COMMITTED` (outras transações podem escrever depois) `... COMMIT`
    - `SET TRANSACTION WRITE ISOLATION LEVEL READ UNCOMMITTED` (pode ler dados não efetivados) `... COMMIT`

Manutenção do Projeto Lógico

- Pode ser necessário para melhoria de desempenho
  - mudança de alternativa de mapeamento
    - exemplo: relacionamento em tabela => relacionamento via chave estrangeira indexada
  - replicação (sem exageros...)
    - exemplo: data do pedido na tabela ItensPedido para facilitar geração de relatório de vendas
  - políticas de particionamento em BDDs
    - tanto horizontal quanto vertical