Optimização do Modelo de Dados

- Conceitos
- Objectivo
- Desnormalização
  - Vantagens vs Desvantagens
  - Processo de Desnormalização
  - Estratégias
- Conclusão

Normalização

- Objectivo
  - Organizar dados
  - Minimizando redundância
  - Minimizando o número de tabelas
  - Maximizando a disponibilidade dos dados
- Problemas
  - Consultas com problemas de desempenho
  - Acesso não optimizado aos dados
  - Caminhos complexos para extrair informação dos sistemas de bases de dados

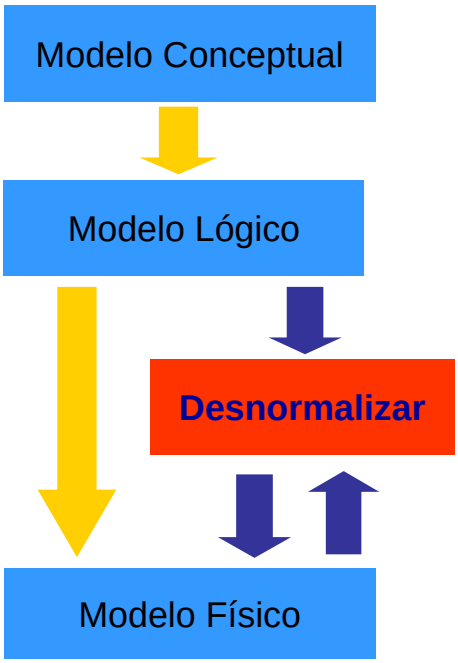
Desnormalização

- Melhorar o desempenho no acesso aos dados armazenados numa Base de Dados normalizada
- Como?
  - Diminuir o número de tabelas físicas e consequentemente minimizar a junção de tabelas
  - Aplicando a **técnica de desnormalização** para a introdução de redundância de modo a melhorar o desempenho
- Porquê?
  - As junções de tabela são operações computacionalmente caras
  - A normalização espalha os dados por inúmeras tabelas, pelo que, extrair informação conduz a comandos SQL (SELECT) com número elevado de tabelas (aumentando consequentemente o número de junções necessárias)

Desnormalização

- Quando?
  - Existem problemas de desempenho na base de dados
- Como?
  - Analisar as aplicações que acedem aos dados e as que podem vir a aceder
- Vantagens vs Desvantagens
  - Redundância de dados
  - Código mais complexo, com regras mais complexas para manter a consistência da informação
  - Sacrificada a flexibilidade
  - Aumentada a velocidade das pesquisas, mas diminui as actualizações

Processo de desnormalização



- Análise de Requisitos
- Identificar/caracterizar entidades e relacionamentos
- Diagrama de Entidade-Relacionamento

Indicar no **modelo lógico**

a) A cardinalidade dos relacionamentos
b) O volume de cada tabela (crescimento num intervalo de tempo)
c) O fluxo (distribuição) dos dados pelas aplicações

A desnormalização não é um processo puramente lógico nem puramente físico

Documentar

a) Situações de desnormalização
b) Efeitos na integridade dos dados
c) Soluções para os efeitos na integridade dos dados

Estratégias

- Reduzir o n.º de tabelas
- Armazenar atributos calculados
- Adicionar colunas redundantes
- Definir vistas e vistas materializadas
- Criar tabelas para substituir vistas
- Dividir uma tabela

Estratégias

- Reduzir o n.º de tabelas
  - Entidades com relacionamento 1:1
  - Entidades com relacionamento 1:N (sem participação obrigatória do lado N)
  - Entidades com relacionamento M:N
  - Hierarquias com aplicação da alternativa C
- Armazenar atributos calculados
- Adicionar colunas redundantes
  - Atributos multivalor
  - Propagar chaves estrangeiras e/outras colunas

Estratégias

- Definir vistas e vistas materializadas
  - Exemplo 1

```
CREATE VIEW v_NadComp
AS
SELECT NOME
FROM TNADADOR N
WHERE EXISTS
          (SELECT *
              FROM TSCORE
              WHERE CODNADADOR=N.CODNADADOR
              AND PCLASSIFICACAO=1);


--Basta consultar a vista
SELECT * FROM v_NadComp;
```

Estratégias

- Definir vistas e vistas materializadas
  - Exemplo 2

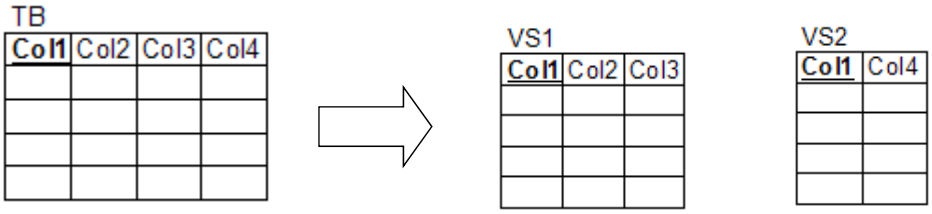
```
CREATE MATERIALIZED VIEW v_NadScore
AS
SELECT CODNADADOR, COUNT(*)
FROM TSCORE
GROUP BY CODNADADOR;

--Basta consultar a vista
SELECT * FROM v_NadScore;

--Actualiza os dados
REBUILD v_NadScore;
```


Estratégias

- Criar tabelas para substituir vistas
  - consultas que envolvem várias tabelas e milhares ou mesmo milhões de registos, podem demorar várias horas, especialmente porque os mesmos dados podem estar a ser alterados durante a própria execução da consulta
- Dividir uma tabela
  - Verificar se algumas colunas/linhas são mais utilizadas que outras
  - Na divisão deve existir uma vista para acesso a todos os dados em todas as tabelas
  - Vertical
  - Horizontal

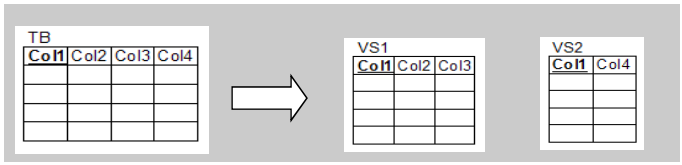
Estratégias

- Dividir uma tabela
  - Vertical
    - frequência de acesso a colunas é muito diferente de coluna para coluna
    - reduz-se o número de páginas/blocos de memória que é necessário ler pois as linhas ocupam menos espaço
    - acesso a todos os dados usando a operação JUNÇÃO
    - Particularmente interessante quando existem colunas que armazenam grandes quantidades de texto. Se esses campos forem raramente acedidos, então, podem ser colocados numa tabela à parte



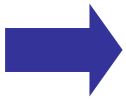
Estratégias

- Dividir uma tabela
  - Vertical
    - Exemplo



in livro *"Database Administration: The Complete Guide to Practices and Procedures"*, C. S. Mullins

```
CREATE TABLE ITEM (
      ItemNum INTEGER,
      ItemSize CHAR(1),
      ItemColor CHAR(10),
      ItemDescr CHAR(100),
      (…)
);
```



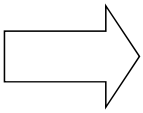
```
CREATE TABLE ITEM (
      ItemNum INTEGER,
      ItemSize CHAR(1),
      ItemColor CHAR(10),
      ItemDescr CHAR(10),
      (…)
);
CREATE TABLE ITEM_DESC (
      ItemNum INTEGER,
      ItemDesc CHAR(90),
      (…)
);
```

Estratégias

- Dividir uma tabela
  - Horizontal
    - divisão dos registos por tabelas consoante a sua classificação, indicada por uma coluna
    - acesso a todos os dados usando a operação UNIÃO

TB

Col1	Col2	Col3	Col4



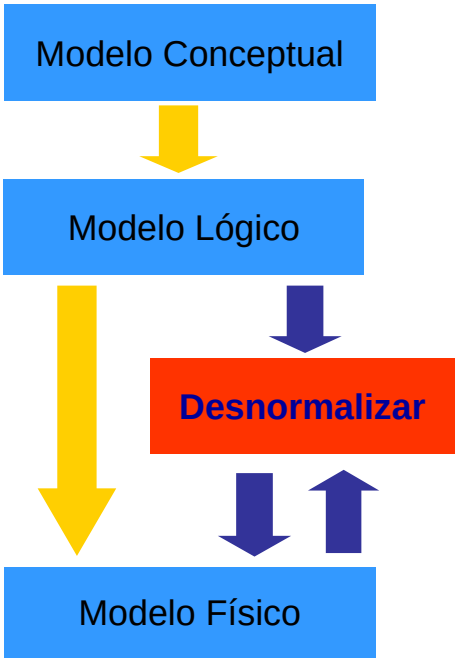
HS1

Col1	Col2	Col3	Col4

HS2

Col1	Col2	Col3	Col4

Processo de desnormalização



- Análise de Requisitos
- Identificar/caracterizar entidades e relacionamentos
- Diagrama de Entidade-Relacionamento

Indicar no **modelo lógico**

a) A cardinalidade dos relacionamentos
b) O volume de cada tabela (crescimento num intervalo de tempo)
c) O fluxo (distribuição) dos dados pelas aplicações

A desnormalização não é um processo puramente lógico nem puramente físico

Documentar

a) Situações de desnormalização
b) Efeitos na integridade dos dados
c) Soluções para os efeitos na integridade dos dados

Conclusão

Bibliografia

- secções 1, 3 e 4 do artigo:

  Denormalization Effects on Performance of RDBMS,

  Lawrence Sanders & Seungkyoon Shin, Procedings of the 34th Hawai International Conference on System Science, 2001

- cap. 4 do livro

  Database Administration: The Complete Guide to Practices and Procedures, C. S. Mullins, Addison-Wesley Pub. Co, 2nd edition (2013)

- cap. 5 do livro

  Oracle Performance Tuning, M. Gurry & P. Corrigan, O'Reilly