

# REFATORAÇÃO EM BANCOS DE DADOS

---

Sérgio Mergen

# Refatoração de Banco de Dados

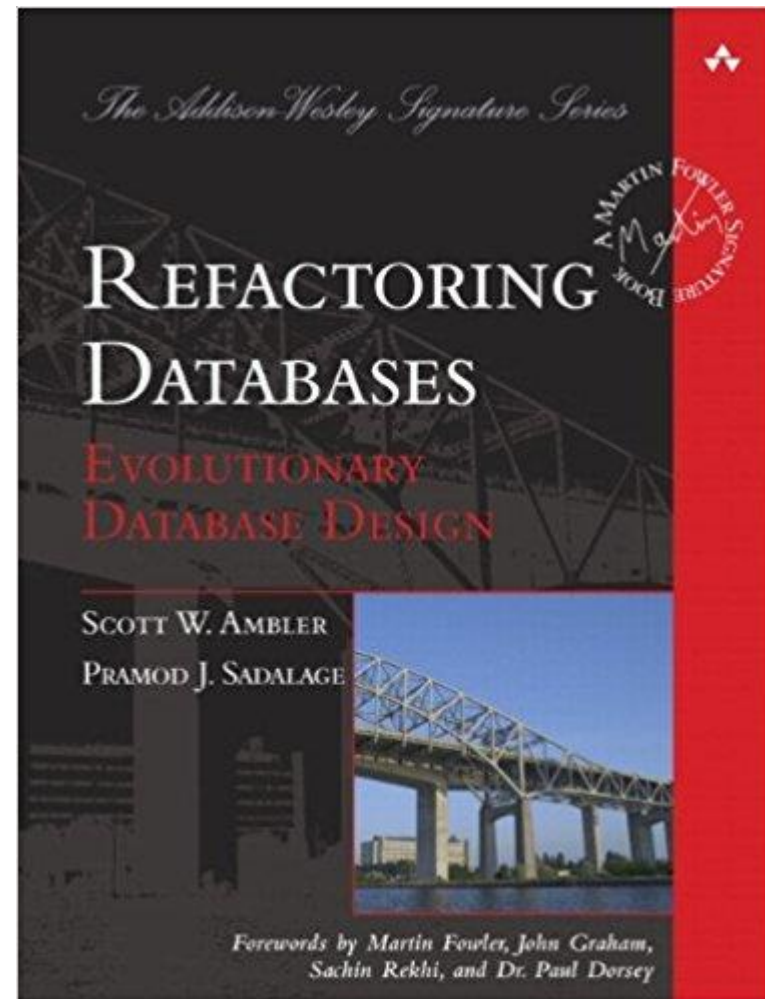
- É o processo de modificar o esquema do banco de dados sem que isso mude a semântica dos dados que estão armazenados
- O conceito é semelhante ao conceito de refatoração de código usado em engenharia de software
  - Só que aplicado a bancos de dados

# Razões para Refatoração

- Em engenharia de software, os maus cheiros no código (code smells) podem indicar que o código precisa ser aprimorado
  - E quanto à bancos de dados?
- Alguns “maus cheiros” em modelos de bancos de dados
  - Colunas com múltiplas finalidades
  - Tabelas com múltiplas finalidades
  - Dados redundantes
  - Tabelas com excessivo número de colunas
  - Tabelas com excessivo número de registros
  - Medo de modificar algo

# Processos de Refatoração

- Existe um catálogo de processos de refatoração.
- Para cada processo, é explicado
  - seu propósito
  - Implementação
  - Exemplos



# Processos de Refatoração

- Nessa aula veremos os seguintes processos
  - Alteração de atributo de coluna
  - Introdução de coluna calculada
  - Criação de tabela detalhe
  - Padronização de Códigos

# Alteração de atributo de coluna

- **Ex.1:** A coluna chamada “descP” contém a descrição de um projeto.
- No entanto, esse nome não é muito intuitivo

Projeto	
*numProj	int
descP	char(30)
Status	char(10)

# Alteração de atributo de coluna

- **Ex.1:** A coluna chamada “descP” contém a descrição de um projeto.
- No entanto, esse nome não é muito intuitivo
- Dessa forma, decidiu-se alterar o nome para “descricao”

Projeto	
*numProj	int
descricao	char(30)
Status	char(10)

# Alteração de atributo de coluna

- Script para alteração de nome da coluna (atributo nome)
- Ex. (alteração de nome em PostgreSQL)  
    ALTER TABLE **projeto**  
        RENAME **descP** TO **descrição**;
- Essa sintaxe é SQL padrão ANSI



# Alteração de atributo de coluna

- Script para alteração de nome da coluna (atributo nome)

- Ex. (alteração de nome em MySQL)

```
ALTER TABLE projeto
```

```
CHANGE descP descricao CHAR (30);
```

- Essa sintaxe não é SQL ANSI
  - Ou seja, nem todos os SGBDs suportam

# Alteração de atributo de coluna

- Script para alteração de nome da coluna (atributo nome)
- Ex. (alteração de nome em MySQL) (**solução alternativa**)
  - Criar uma nova coluna (**descricao**)
  - Migrar valores de **descP** para **descricao**
  - Remove coluna **descP**
- Esse é um processo de refatoração conhecido como deslocamento de colunas
  - Veremos em outra aula

# Alteração de atributo de coluna

- **Ex.2:** A coluna status é uma informação opcional.

Projeto		
*numProj	int	not null
descP	char(30)	not null
dataIni	date	not null
Status	char(10)	null

# Alteração de atributo de coluna

- **Ex.2:** A coluna status é uma informação opcional.
- Devido a mudança de requisitos, ela passará a ser obrigatória

Projeto		
*numProj	int	not null
descP	char(30)	not null
dataIni	date	not null
Status	char(10)	not null

# Alteração de atributo de coluna

- Script para alteração de obrigatoriedade de coluna
  - Passo 1: atualização de valores nulos
  - Passo 2: alteração de obrigatoriedade

# Alteração de atributo de coluna

- Script para alteração de obrigatoriedade de coluna
  - Passo 1: atualização de valores nulos
- Ex.  

```
UPDATE projeto  
SET status = 'sem status' WHERE status IS NULL;
```
- Todos registros devem ter valores não nulos para que a restrição possa ser criada
  - Nesse passo, todos valores nulos são removidos

# Alteração de atributo de coluna

- Script para alteração de obrigatoriedade de coluna
  - Passo 2: alteração de obrigatoriedade
- Ex. (alteração de obrigatoriedade em PostgreSQL)  
ALTER TABLE **projeto**  
ALTER COLUMN **status SET NOT NULL;**
- Essa sintaxe é SQL ANSI

# Alteração de atributo de coluna

- Script para alteração de obrigatoriedade de coluna
  - Passo 2: alteração de obrigatoriedade
- Ex. (alteração de obrigatoriedade em MySQL)  
ALTER TABLE **projeto**  
MODIFY **status** CHAR (10) NOT NULL;
- Essa sintaxe não é SQL ANSI
  - O formato pode mudar de banco para banco



# Alteração de atributo de coluna

- Tipos de dados também podem ser trocados
  - Ex. trocar de INT para BIGINT
  - ALTER TABLE **projeto** MODIFY **numProj** BIGINT;
- Os valores são convertidos para o tipo de dados novo
- Cuidado
  - Pode-se perder informações na conversão para tipos de dados mais específicos
  - Ex.
    - Int para smallint
    - Datetime para date
    - Char(50) para char(10)
- Nesses casos, corre-se o risco de ter os valores “truncados”

# Alteração de atributo de coluna

- A troca para tipos de dados incompatíveis também é possível
  - Ex. trocar de date para char(8)
  - ALTER TABLE **projeto** MODIFY **dataIni** CHAR(8);
- No entanto, é recomendável que antes se verifique a possibilidade de perda de informação.
  - '2015-09-01' ficaria '2015-09-' e não '20150901'
- Caso seja possível ocorrer perda, é melhor utilizar o processo de deslocamento de coluna
  - Criar uma coluna nova
  - Migrar dados para a coluna nova
  - Remover a coluna antiga
- Veremos esse processo em outra aula

# Alteração de atributo de coluna

- **Ex. 3:** A data de término de um projeto (dataFim) deve ser superior a data do seu início (dataIni).
- Porém, essa regra não é validada por algumas aplicações que adicionam projetos

Projeto	
*numProj	int
descP	char(30)
dataIni	date
dataFim	date

# Alteração de atributo de coluna

- **Ex. 3:** A data de término de um projeto (dataFim) deve ser superior a data do seu início (dataIni).
- Porém, essa regra não é validada por algumas aplicações que adicionam projetos
- Dessa forma, decidiu-se criar essa **restrição de integridade** no próprio SGBD

Projeto	
*numProj	int
descP	char(30)
dataIni	date
dataFim	date

# (ABRE PARÊNTESES) RESTRIÇÕES DE INTEGRIDADE

---

# Tipos de Restrições de integridade

- Integridade de vazio
  - O valor da coluna não pode ser vazio
  - Ex. colunas NOT NULL
- Integridade de chave
  - O valor da coluna não pode se repetir dentro da tabela
  - Ex. colunas chave primária ou colunas UNIQUE
- Integridade Referencial
  - O valor da coluna deve obrigatoriamente aparecer como chave primária em outro lugar
  - Ex. colunas que pertençam a uma chave estrangeira

# Tipos de Restrições de integridade

- Integridade de domínio
  - A valor da coluna deve pertencer a um conjunto de valores predefinido
  - Ex. status = 'ativo' ou 'inativo'
- Integridade semântica
  - O valor da coluna deve respeitar regras de negócio
  - Ex. dataFim > dataIni
- A forma de criação dessas restrições de integridade depende do SGBD utilizado

# (FECHA PARÊNTESES) RESTRIÇÕES DE INTEGRIDADE

---



# Alteração de atributo de coluna

- Script para criação de restrição de integridade
  - Uma das formas de criar essa restrição é através de CHECK CONSTRAINTS
- Ex. (postgres)  

```
ALTER TABLE projeto  
    ADD CONSTRAINT check_data  
    CHECK (dataIni < dataFim );
```
- Nem todos bancos suportam check constraints
- Nesse caso, essa restrição teria que ser validada de outra forma
  - Ex. Via Trigger
  - Veremos em outra aula

# Alteração de atributo de coluna

- A alteração de atributo de uma coluna que contenha alguma restrição de integridade referencial nem sempre é fácil
  - A coluna pode pertencer a uma chave primária
  - A coluna pode pertencer a um relacionamento de chave estrangeira

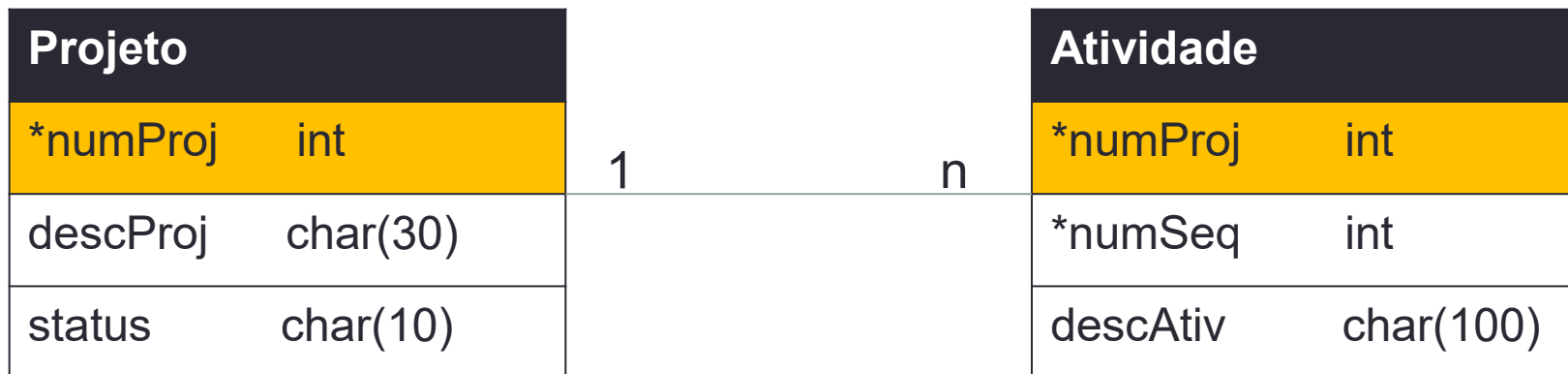
# Alteração de atributo de coluna

- Se a coluna for parte de chave primária
  - Em alguns casos é necessário recorrer ao processo de deslocamento de coluna
  - Veremos em outra aula
- Ex. Passar numProj para bigInt

Projeto	
*numProj	int
descProj	char(30)
status	char(10)

# Alteração de atributo de coluna

- Se a coluna for parte de um relacionamento de chave estrangeira
  - Geralmente o atributo precisa ser igual nas duas colunas envolvidas (chave primária e chave estrangeira)
  - Ou seja, esse processo teria que ser propagado até outras tabelas
- Ex. Passar numProj de Projeto para bigInt



# Processos de Refatoração

- Nessa aula veremos os seguintes processos
  - Alteração de atributo de coluna
  - **Introdução de coluna calculada**
  - Criação de tabela detalhe
  - Padronização de Códigos

# Introdução de coluna calculada

- O total de horas gastas em cada projeto é uma informação bastante requisitada



# Introdução de coluna calculada

- O total de horas gastas em cada projeto é uma informação bastante requisitada
- Desse modo, pretende-se armazenar o total junto ao projeto
  - Assim, pode-se acessar esse valor diretamente
  - Em vez de recorrer a uma consulta com agrupamentos

PROJETO	
*idProj	int
NomeProj	char(30)
Total	int

1

n

ATIVIDADE	
*idProj	int
*num_seq	int
Data	date
Desc	char(50)
Horas	int

# Introdução de coluna calculada

- Script para introdução de coluna calculada
  - Passo 1: criação e alimentação da coluna calculada
  - Passo 2: criação da forma de atualização automática



# Introdução de coluna calculada

- Script para introdução de coluna calculada
  - Passo 1: criação e alimentação da coluna calculada

Ex.

//criação da coluna

```
ALTER TABLE projeto ADD COLUMN total INT;
```

//alimentação da coluna calculada

```
UPDATE projeto p SET total = SELECT SUM (horas)  
FROM atividade a WHERE p.idProj = a.idProj;
```

# Introdução de coluna calculada

- Script para introdução de coluna calculada
  - Passo 2: criação da forma de atualização automática

Ex. (usando trigger)

//criação da forma de atualização via trigger

```
CREATE TRIGGER trigger_total_projeto AS (...);
```

- A trigger deve ser disparada sempre que for adicionado um registro na tabela “atividade”.
  - Falaremos mais sobre triggers em outra aula.
- Alternativa: Controlar as atualizações via aplicação
  - **Importante:** **todas** aplicações que adicionam atividades devem atualizar o total

# Processos de Refatoração

- Nessa aula veremos os seguintes processos
  - Alteração de atributo de coluna
  - Introdução de coluna calculada
  - Criação de tabela detalhe
  - Padronização de Códigos

# Criação de tabela detalhe

- Uma empresa possui cadastro de fornecedores internacionais.
- O DBA acha melhor que seja possível gerenciar os países com quem a empresa possui parcerias.

Fornecedor	
*idForn	int
nome	char(50)
país	char(30)

# Criação de tabela detalhe

- Uma empresa possui cadastro de fornecedores internacionais.
- O DBA acha melhor que seja possível gerenciar os países com quem a empresa possui parcerias.
- Dessa forma, será criada uma tabela detalhe para guardar os países



# Criação de tabela detalhe

- Script para criação de tabela detalhe
  - Passo 1: criação da tabela detalhe
  - Passo 2: alimentação da tabela detalhe
  - Passo 3: atualização de valores da tabela original
  - Passo 4: criação de restrição de chave estrangeira
  - Passo 5: remoção de colunas obsoletas

# Criação de tabela detalhe

- Script para criação de tabela detalhe
  - Passo 1: criação da tabela detalhe
- Ex.

```
CREATE TABLE pais (  
    idPais    int AUTO_INCREMENT,  
    nome      char(30),  
    PRIMARY KEY (idPais)  
);
```

- Com o auto incremento, é possível criar registros de pais usando apenas a informação do seu nome.

# Criação de tabela detalhe

- Script para criação de tabela detalhe
  - Passo 2: alimentação da tabela detalhe
- Ex.

```
INSERT INTO pais (nome)  
        SELECT DISTINCT pais FROM fornecedor;
```

- O DISTINCT garante que não se tentará gerar registros com siglas duplicadas
- Cada nome distinto será associado a uma chave primária sequencial gerada automaticamente



# Criação de tabela detalhe

- Script para criação de tabela detalhe
  - Passo 3: atualização de valores da tabela original
- Ex.

```
ALTER TABLE fornecedor  
    ADD COLUMN idPais int;
```

```
UPDATE fornecedor f SET idPais =  
    SELECT DISTINCT idPais FROM pais p  
    WHERE p.nome = f.pais
```

- A coluna contendo o nome do país é usada para encontrar o valor correspondente à chave primária desse país.

# Criação de tabela detalhe

- Script para criação de tabela detalhe
  - Passo 4: criação de restrição de chave estrangeira
- Ex.

```
ALTER TABLE fornecedor  
  ADD FOREIGN KEY 'fk_forn_pais'  
    (idPais) REFERENCES pais (idPais);
```

- Esse passo só pode ser executado depois que a coluna **idPais** em **fornecedor** tiver sido atualizada

# Criação de tabela detalhe

- Script para criação de tabela detalhe
  - Passo 5: remoção de colunas obsoletas
- Ex.

```
ALTER TABLE fornecedor  
    DROP COLUMN pais;
```

- A coluna **pais** em **fornecedor** não é mais necessária.


# Processos de Refatoração

- Nessa aula veremos os seguintes processos
  - Alteração de atributo de coluna
  - Introdução de coluna calculada
  - Criação de tabela detalhe
  - Padronização de Códigos

# Padronização de Códigos

- O status de projetos é um campo de texto livre, que pode ser preenchido com qualquer valor.
- Para facilitar o uso efetivo dessa informação em consultas e relatórios, convencionou-se que apenas valores predeterminados podem ser usados

Projeto	
*idProj	int
nome	char(50)
Status	char(20)




1, 'Proj A', 'concluído'  
2, 'Proj B', 'aberto'  
3, 'Proj C', 'encerrado'  
4, 'Proj D', 'não encerrado'

# Padronização de Códigos

- O status de projetos é um campo de texto livre, que pode ser preenchido com qualquer valor.
- Para facilitar o uso efetivo dessa informação em consultas e relatórios, convencionou-se que apenas valores pre-determinados podem ser usados
- Dessa forma, faz-se necessário migrar os valores 'sujos' de status para valores bem comportados

Projeto	
*idProj	int
nome	char(50)
Status	char(20)



1, 'Proj A', 'encerrado'  
2, 'Proj B', 'aberto'  
3, 'Proj C', 'encerrado'  
4, 'Proj D', 'aberto'

# Padronização de Códigos

- Script para padronização de códigos
  - Passo 1: atualização de valores
  - Passo 2: criar restrição de integridade

# Padronização de Códigos

- Script para padronização de códigos

- Passo 1: atualização de valores

- Ex.

UPDATE **projeto** SET **status** = 'Encerrado' WHERE **status** = 'concluído';

UPDATE **projeto** SET **status** = 'Aberto' WHERE **status** = 'não encerrado';

- Pode-se usar LIKE para criar comandos de atualização mais abrangentes

- Ex. WHERE **status** LIKE 'não enc%'

- O operador IN pode atingir o mesmo propósito

- Ex. WHERE **status** IN ('não encerrado', 'não concluído', 'em execução')



# Padronização de Códigos

- Script para padronização de códigos
  - Passo 2: criar restrição de integridade

- Ex. (postgres)

```
ALTER TABLE projeto
```

```
ADD CONSTRAINT check_status
```

```
CHECK (status IN ('Aberto', 'Encerrado') );
```

- Essa restrição garante que não se consiga inserir valores inválidos
- E quanto à SGBDs que não suportam CHECK CONSTRAINTS ?


# Padronização de Códigos

- Script para padronização de códigos
  - Passo 2: criar restrição de integridade
- Ex. (mysql) (solução alternativa)  
ALTER TABLE **projeto**  
MODIFY status ENUM ('Aberto', 'Encerrado') );
- Outras soluções
  - Uso de triggers
  - Criação de uma tabela detalhe

# Atividade Individual

- A tabela Projeto possui uma coluna para indicar o status do projeto.
- Decidiu-se que essa informação passaria a ser gerenciada por uma tabela detalhe
- No entanto, os valores de status guardados não estão padronizados

Projeto	
*idProj	int
nome	char(50)
Status	char(20)




1, 'Proj A', 'concluído'  
2, 'Proj B', 'nao aberto'  
3, 'Proj C', 'aberto'  
4, 'Proj D', 'nao aberto'  
...

# Atividade Individual

- Crie um script de refatoração que
  - gere uma tabela para armazenar os dois status de projeto possíveis
    - Aberto
    - Encerrado
  - Altere a tabela projeto para que ela passe a usar a tabela criada

Projeto	
*idProj	int
nome	char(50)
Status	char(20)




1, 'Proj A', 'concluído'  
2, 'Proj B', 'nao aberto'  
3, 'Proj C', 'aberto'  
4, 'Proj D', 'nao aberto'  
...

# Atividade Individual

- Etapas
  - Padronização dos códigos
  - Criação da tabela detalhe
  - Alimentação da tabela detalhe
  - Criação da fk em projeto
  - Alimentação da fk em projeto
  - Remoção da coluna 'status' em projeto

Projeto	
*idProj	int
nome	char(50)
Status	char(20)




1, 'Proj A', 'concluído'  
2, 'Proj B', 'nao aberto'  
3, 'Proj C', 'aberto'  
4, 'Proj D', 'nao aberto'  
...

# Atividade Individual

- Para a etapa de padronização, não ajuste os valores manualmente.
- Faça a padronização através de código SQL

Projeto	
*idProj	int
nome	char(50)
Status	char(20)



1, 'Proj A', 'concluído'  
2, 'Proj B', 'nao aberto'  
3, 'Proj C', 'aberto'  
4, 'Proj D', 'nao aberto'  
...