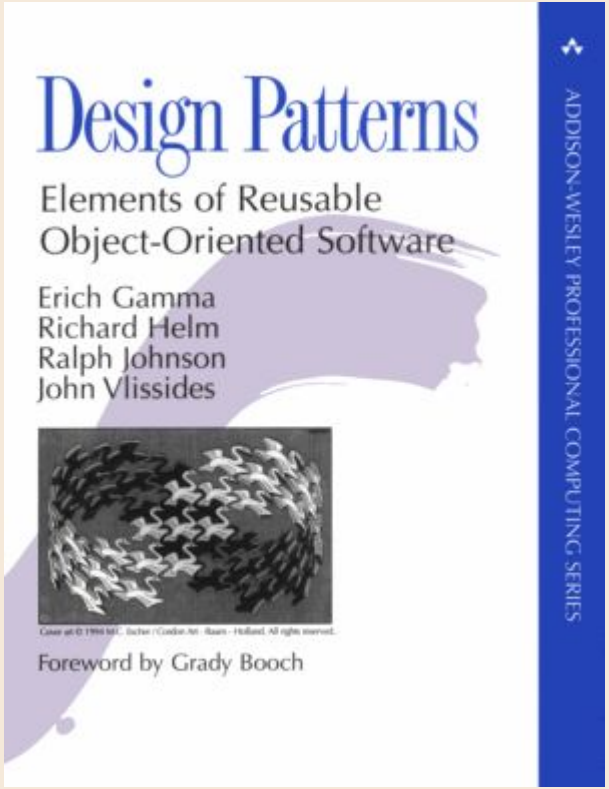
Modelagem

Anti-patterns

Anti-pattern

Algum padrão repetido de ação, processo ou estrutura que inicialmente parecia benéfico, mas, como consequência produz mais malefícios do que benefícios



O termo foi cunhado em 1995 por Andrew Koenig, inspirado pelo livro do Gang of Four Design Patterns

Normalização de banco de dados

- **Edgar F. Codd**, definiu três dessas formas:

1 - NF

- Atributos precisam ser atômicos, o que significa que as tabelas não podem ter valores repetidos e nem atributos possuindo mais de um valor.
- Ex: CLIENTE = {ID + ENDEREÇO + TELEFONES}

2 - NF

- Atributos normais, ou seja, os não chave, devem depender unicamente da chave primária da tabela.
- Ex: PROFESSOR_CURSO = {ID_PROF + ID_CURSO + SALARIO + DESCRICAO_CURSO}

3 - FN

- define que todos os atributos dessa tabela devem ser funcionalmente **independentes** uns dos outros, ao mesmo tempo que devem ser dependentes exclusivamente da chave primária da tabela
- Ex: FUNCIONARIO = {ID + NOME + **VALOR_SALARIO** + **VALOR_FGTS**}

Básico

- Use **BONS** nomes
    - Se não conseguir, talvez você não precise ou não entendeu
- Use NOT NULL
- Siga algum padrão
- Tabelas com “**Single responsibility**” (BySOLID)

```
transaction_datetime_local timestamp without time zone
transaction_datetime timestamp without time zone
```

- Usando o mesmo nome de coluna em tabelas diferentes, mas com tipos de dados diferentes
- Aninhando VIEWs como se fossem bonecas russas
- Abreviações
- Usando SELECT DISTINCT para mascarar um problema de junção

Exemplos

Bad Smells

- **Multi-purpose column**
- **Multi-purpose table (God tables)**
  - Geralmente é o resultado de uma tentativa de encapsular uma grande parte dos dados para um domínio de negócio em uma única tabela

- EAV
- Usar o tipo do dado correto

```
-- Entity Attribute Values
CREATE TABLE eav
(
    id bigserial PRIMARY KEY,
    entity    text NOT NULL,
    entity_id text NOT NULL,
    parameter text NOT NULL,
    value     text NOT NULL,
    UNIQUE (entity, entity_id, parameter)
);
```

Multi-purpose column

```sql
CREATE TABLE pessoa (
    id    BIGSERIAL PRIMARY KEY,
    tipo  TEXT CHECK (tipo IN ('FISICA', 'JURIDICA')),
    nome  TEXT NOT NULL,
    data  DATE
);
```

```java
Pessoa pessoa = PessoaRepository.findById(10L);
PessoaDTO pessoaDTO = new PessoaDTO();

if (pessoa.IsFisica()) {
    pessoaDTO.setDataNascimento(pessoa.getData());
} else {
    pessoaDTO.setDataFundacao(pessoa.getData());
}
```

Refactor: Multi-purpose column

- Split Column
- Move Column
- Split table

Multi-purpose table

```sql
CREATE TABLE pessoa (
    id    BIGSERIAL PRIMARY KEY,
    tipo  TEXT CHECK (tipo IN ('FISICA', 'JURIDICA')),
    nome          TEXT NOT NULL,
    nome_fantasia TEXT,
    cnpj          VARCHAR(14),
    cpf           VARCHAR(11),
    rg            VARCHAR(10),
    data          DATE
);
```

Multi-purpose table

```
Pessoa pessoa = PessoaRepository.findById(10L);
PessoaDTO pessoaDTO = new PessoaDTO();

if (pessoa.IsFisica()) {
    pessoaDTO.setNome(pessoa.getNome());
    pessoaDTO.setCPF(pessoa.getCPF());
    pessoaDTO.setRG(pessoa.getRG());
    pessoaDTO.setDataNascimento(pessoa.getData());
} else {
    pessoaDTO.setRazaoSocial(pessoa.getNome());
    pessoaDTO.setNomeFantasia(pessoa.getNomeFantasia());
    pessoaDTO.setCNPJ(pessoa.getCNPJ());
    pessoaDTO.setDataFundacao(pessoa.getData());
}
```

Refactor: Multi-purpose table

- Split Column
- Move Column
- Split Table

ENUM

```
CREATE TABLE product
(
    id serial primary key,
    name text not null,
    type text not null
);

INSERT INTO product (name, type) VALUES ('Pizza', 'FOOD');
INSERT INTO product (name, type) VALUES ('Coca-cola', 'DRINK');

-- OR :(

INSERT INTO product (name, type) VALUES ('Coca-cola', 'foo');
```

```
CREATE TYPE product_type AS ENUM('FOOD', 'DRINK', 'DESSERT');

CREATE TABLE product
(
    id serial primary key,
    name text not null,
    type product_type not null
);

INSERT INTO product (name, type) VALUES ('Coca-cola', 'foo');

ERROR:  22P02: invalid input value for enum product_type: "foo"
LINE 1: ...NSERT INTO product (name, type) VALUES ('Coca-cola', 'foo');
```

Por que a refatoração de banco de dados é difícil

- Aplicação
- Modificar o esquema do banco de dados
- Migração de Dados
- Atualizar programas externos
- Execute seus testes
- **Dê pequenos passos**

Desnormalização

- Performance
    - Muitas vezes você não precisa
- Sabedoria para saber quando
- Funcionalidade muito genérica
- ORM e facilidade

```
@Entity
public class Item {

    ...

    @ManyToOne(fetch = FetchType.EAGER)
    private User seller;

    @OneToMany(fetch = FetchType.EAGER)
    private Set<Bid> bids = new HashSet<Bid>();

    ...
}
```

Obrigado!

Lucas Viecelli
lucas.viecelli@ifood.com.br
telefone 49-99194-4888

Referência

https://martinfowler.com/books/refactoring.html

http://www.ambysoft.com/books/refactoringDatabases.html

https://www.slideshare.net/fabriziomello/bad-smells-mal-cheiros-em-bancos-de-dados-81028063