

Testes Automatizados com Banco de Dados

Helves Domingues e Paulo Cheque

12/02/2009 Verão2009

AgilCoop 



Por quê testar BDs?

- Dados => \$
- Lógica
 - Stored Procedures
 - Triggers
- SQL
- Design => Manutenção e Evolução
- Mapeamento ORM
- Segurança

Por quê testar BDs?

- O que é banco de dados evolutivos?
 - Modelagem de dados evolutiva
 - Testes do banco de dados
 - Gerência de configuração (SVN, CVS, GIT, etc)
 - Refatoração do banco de dados

Preocupações

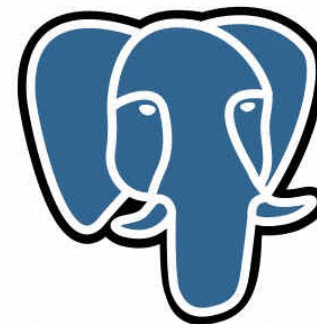
- Compartilhamento da base
- Permissões: Schemas, CRUD
- Execução concorrente do teste
- Muitos sistemas utilizando a mesma base
- Sistemas legados
- Conexão – Rede - Distribuição: Desempenho
- Replicação
- Auditoria

Ambiente de Teste

- Bancos de Dados Compartilhados
- Bancos de Dados Locais
- Bancos de Dados em Memória



PostgreSQL



ORACLE®

Bancos de Dados Compartilhados

- Vantagens:
 - Dados reais de usuários
 - Ambiente mais próximo do real
 - Dados já existentes: Diversas situações prontas
- Desvantagens:
 - Ambiente não controlado para testes
 - Problemas com permissões
 - Preocupação com dados existentes
 - Preocupação com dados criados pelos testes
 - Lentidão

Bancos de Dados Locais

- Vantagens:

- Ambiente controlado para testes
- Melhor desempenho

- Desvantagens:

- Custo de preparação dos dados
- Querys adicionais
- Dados criados por desenvolvedores, não usuários
- Testes de aceitação?
- Preocupação com dados criados pelos testes
- Licenças

Bancos de Dados em Memória

- Vantagens:

- Desempenho
- Ambiente controlado para testes

- Desvantagens:

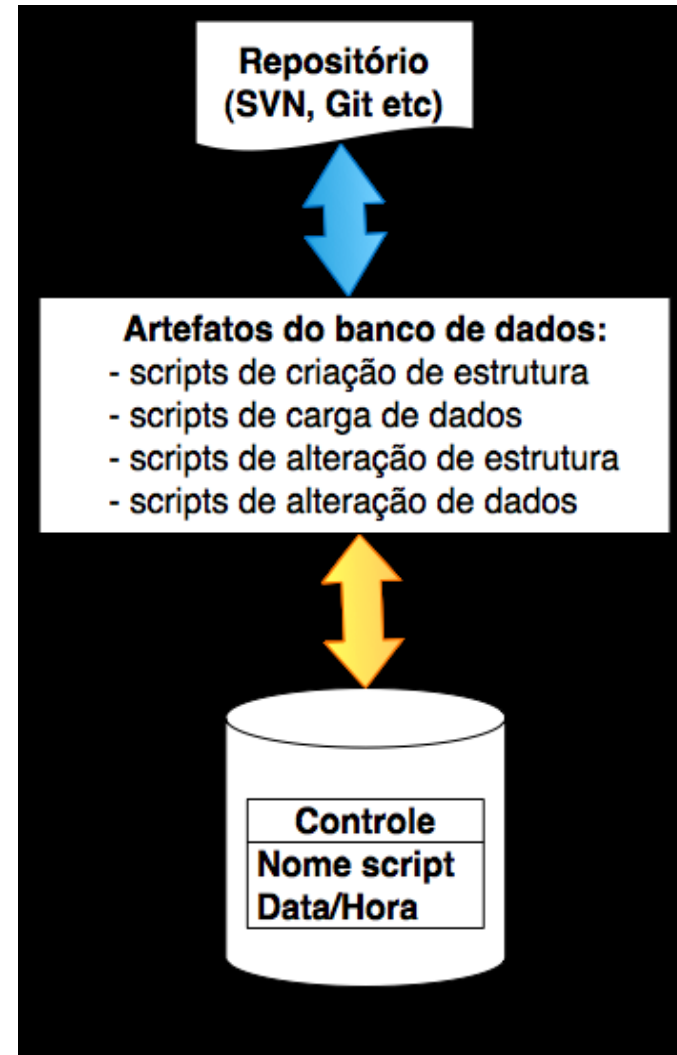
- Ambiente diferente da realidade
- Custo de preparação dos dados
- Querys adicionais
- Dados criados por desenvolvedores, não usuários
- Testes de aceitação?
- Limitações: Stored Procedures, Triggers...

Perguntas

- É possível escolher o ambiente de testes?
- O que é necessário testar?
- Onde estão os erros?
- Qual o custo das desvantagens e o benefício das vantagens?

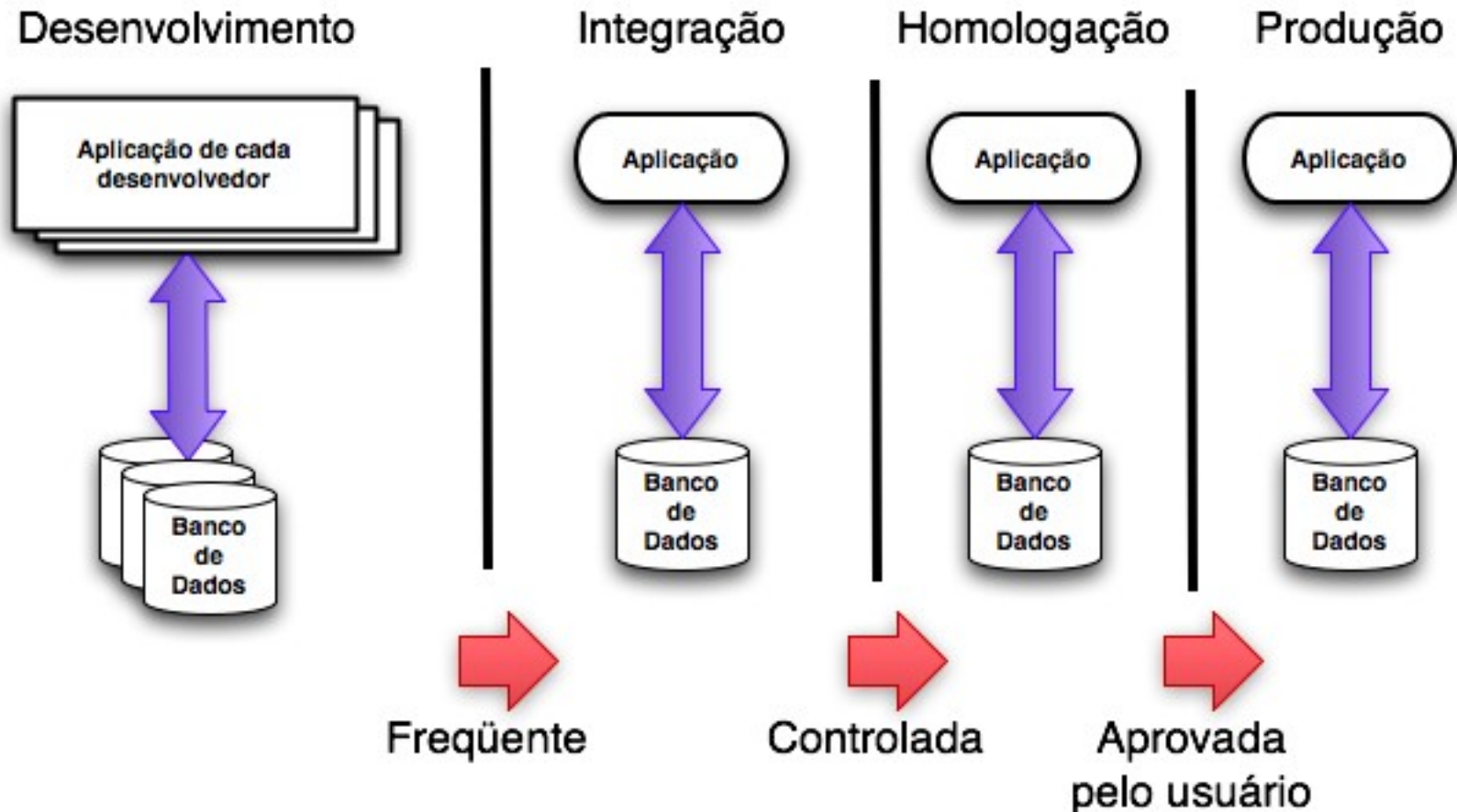
Estratégias

- Ambiente do Desenvolvedor:
 - Controlador de versões
 - Scripts
 - Banco de Dados individual



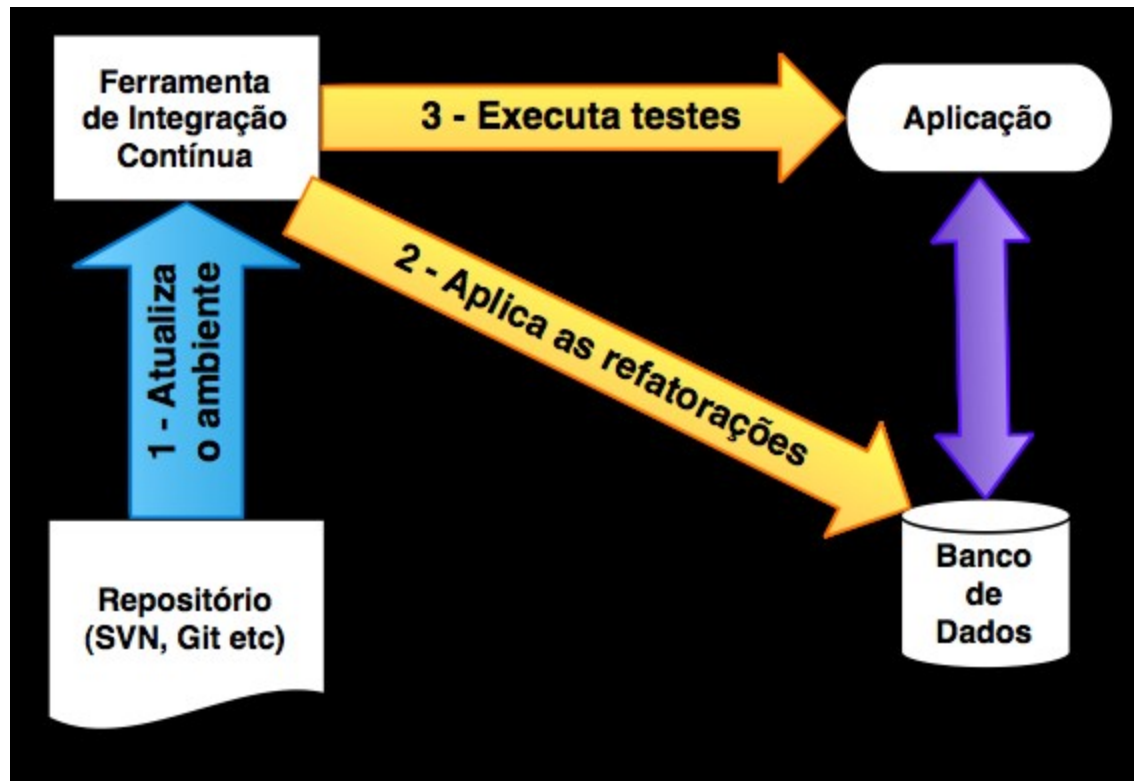
Estratégias

- Ambientes, bancos de dados e transições



Estratégias

- Transição para o ambiente de Integração



Padrões

- Organização:

- Testcase Superclass

- Ex: GrailsUnitTestCase

- Test Utility Method

- Test Helper

```
@After  
public static void removeTodosRegistrosDoBanco() {  
    truncateTabelas();  
}
```

- Database Sandbox: 1 por desenvolvedor

- Transaction Rollback Teardown

- Table Truncation Teardown

+ Padrões

- Generated Values
 - IDs
 - Timestamp
 - Aleatórios

```
new Pessoa(IDGenerator.uniqueID()).save()
```

```
CreditCardGenerator.generate(CreditCardFactory.getMasterCard())
```

Anti-padrões

- Testes frágeis
- Dados/Ambientes não controlados
- IDs fixos
- Lentos
- Sujeira na base de dados

Anti-padrões

```
-
3 class VolumeIntegrationTests extends grails.test.GrailsUnitTestCase {
4
5     void testToString() {
6         def volume = new Volume(largura: 1.44, comprimento: 2.45, altura: 3.46)
7         assertEquals("1.44m x 2.45m x 3.46m", volume.toString())
8         volume.save(flush: true)
9         assertEquals("1.44m x 2.45m x 3.46m", Volume.get(1).toString())
10        assertEquals("1.44m x 2.45m x 3.46m", Volume.findAll().get(0).toString())
11    }
12
13    void testAceitaNoMaximo2CasasDecimal() {
14        def volume = new Volume(largura: 1.441, comprimento: 2.445, altura: 3.449)
15        volume.save(flush: true)
16        assertEquals("1.44m x 2.45m x 3.45m", Volume.get(2).toString())
17        assertEquals("1.44m x 2.45m x 3.45m", Volume.findAll().get(0).toString())
18    }
19    --
```


Sistema Sob Teste

- ... Database Connectivity (DBC)
 - ODBC, JDBC ...
- Frameworks para queries
 - Spring-JDBC ...
- Object-Relational Mapping
 - Hibernate
 - ActiveRecord
- Stored Procedures - Triggers

DBC

- Erros de strings
 - Conversão
 - Valores nulos
- Pouco encapsulamento
 - Código repetido
- Conexão
- Transações

```

8 public class UsuarioDAO {
9     JdbcTemplate jdbc;
10
11 public UsuarioDAO(JdbcTemplate jdbc) {
12     this.jdbc = jdbc;
13 }
14
15 public void insereUsuario(String nome, String senha) {
16     int id = jdbc.queryForInt("select max(id) from usuario");
17     id += 1;
18     Object[] args = new Object[] {id, nome, senha};
19     jdbc.update("insert into usuario (id, nome, senha) values (?, ?, ?)", args);
20 }
21
22 public void atualizaUsuario(Integer id, String nome, String senha) {
23     jdbc.update("update usuario set nome = ?, senha = ? where id = ?", new Object[] {nome, senha, id});
24 }
25
26 public void removeUsuario(String nome) {
27     jdbc.update("delete from usuario where nome = ?", new Object[]{nome});
28 }
29
30 @SuppressWarnings("unchecked")
31 public List<Map<String, Object>> buscaPorNome(String nome) {
32     List<Map<String, Object>> map;
33     map = jdbc.queryForList("select * from usuario where nome like ?", new Object[]{nome.replaceAll("[*]", "%")});
34     return map;
35 }
36 }

```

```
52 @Test
53 public void testaCRUD() {
54     // INSERT
55     dao.insereUsuario("Fulano", "123456");
56     // VERIFICO
57     assertEquals(1, dao.buscaPorNome("Fulano").size());
58     assertEquals("123456", dao.buscaPorNome("Fulano").get(0).get("SENHA"));
59
60     // UPDATE
61     dao.atualizaUsuario(1, "Ciclano", "123457");
62     // VERIFICO
63     assertEquals(1, dao.buscaPorNome("Ciclano").size());
64     assertEquals("123457", dao.buscaPorNome("Ciclano").get(0).get("SENHA"));
65
66     // DELETE
67     dao.removeUsuario("Fulano");
68     assertEquals(0, dao.buscaPorNome("Fulano").size());
69 }
70
71 @Test
72 public void testaBuscaPorNome() {
73     dao.insereUsuario("Fulano", "123456");
74     dao.insereUsuario("Fulano Tal", "123456");
75     dao.insereUsuario("Ciclano Tal", "123456");
76
77     // Ambiente controlado!
78     assertEquals(1, dao.buscaPorNome("Fulano").size());
79     assertEquals(2, dao.buscaPorNome("Fulano*").size());
80     assertEquals(2, dao.buscaPorNome("Fulano%").size());
81     assertEquals(0, dao.buscaPorNome(" Fulano").size());
82     assertEquals(3, dao.buscaPorNome("*lano*").size());
83 }
84
```

Object-Relational Mapping (ORM)

- Tipos
- Validações
 - Min, max, range, null, vazios, padrão, precisão
- Conversões: Tipos primitivos ou objetos
- Metadados: XML, Anotações, Convenções ...

```
1 class Endereco {
2     String cep
3
4     static constraints = {
5         cep(size: 9..9, blank: false, unique: false, matches: "[0-9]{5}-[0-9]{3}")
6     }
```

```
3 class EnderecoIntegrationTests extends grails.test.GrailsUnitTestCase {
4
5     private Endereco criaEnderecoComCEP(String cep) {
6         def endereco = new Endereco(
7             cep: cep,
8             rua: "Av. Paulista",
9             numero: 1,
10            complemento: "Bloco B / Sala 3",
11            bairro: "Vergueiro",
12            cidade: "São Paulo",
13            estado: "São Paulo",
14            pais: "Brasil"
15        )
16        endereco
17    }
18
19    void testCEPValido() {
20        assertTrue(criaEnderecoComCEP("12345-123").validate())
21    }
22
23    void testCEPInvalido() {
24        assertFalse(criaEnderecoComCEP(null).validate())
25        assertFalse(criaEnderecoComCEP("").validate())
26        assertFalse(criaEnderecoComCEP("12345123").validate())
27        assertFalse(criaEnderecoComCEP("1-2345123").validate())
```

```

1 class Volume { // em metros
2     Float largura
3     Float comprimento
4     Float altura
5
6     static belongsTo = [ Imovel ]
7
8     static constraints = {
9         largura(min: 0f, max: 9999f, scale: 2, nullable: false)
10        comprimento(min: 0f, max: 9999f, scale: 2, nullable: false)
11        altura(min: 1f, max: 9999f, scale: 2, nullable: true)
12    }
13
14    String toString() {
15        if(altura)
16            largura + "m x " + comprimento + "m x " + altura + "m"
17        else
18            largura + "m x " + comprimento + "m"
19    }
20
21    Float area() {
22        largura * comprimento
23    }
24
25    Float total() {
26        if(altura)
27            largura * comprimento * altura
28        else
29            area()
30    }
31 }

```

```

-
3 class VolumeIntegrationTests extends grails.test.GrailsUnitTestCase {
4
5     void testToString() {
6         def volume = new Volume(largura: 1.44, comprimento: 2.45, altura: 3.46)
7         assertEquals("1.44m x 2.45m x 3.46m", volume.toString())
8         volume.save(flush: true)
9
10        assertEquals("1.44m x 2.45m x 3.46m", Volume.findAll().get(0).toString())
11    }
12
13    void testAceitaNoMaximo2CasasDecimal() {
14        def volume = new Volume(largura: 1.441, comprimento: 2.445, altura: 3.449)
15        volume.save(flush: true)
16
17        assertEquals("1.44m x 2.45m x 3.45m", Volume.findAll().get(0).toString())
18    }
--

```


ORM: Mapeamentos

- Mapeamentos:
 - 1-1, 1-N, N-N

- Herança:
 - Tabela única
 - Tabela por tipo

- Listas, mapas ...

ORM: Facilidades

- Cascata de Insert, Update e Delete
- Carregamento de objetos:
 - Preguiçosa, ansiosa
- Cache
- Eventos
 - Before/After: insert / update / delete / load

```
void testCascataDeCaracteristicasDeUmImovel() {
    // Prepara
    def imovel = criaImovelValido()
    imovel.addToCaracteristicas(new CaracteristicaDoImovel(caracteristica: mobiliado, quantidade:1))
    assertTrue("Ops, imóvel inválido!", imovel.validate())

    // Executa INSERT
    imovel.save(flush: true)

    // Verifica
    def imovelConsultado = Imovel.findAll().get(0)
    assertNotNull(imovelConsultado)
    assertEquals(1, imovelConsultado.caracteristicas.size())

    // Verifica cascatas
    assertEquals(1, Localizacao.findAll().size())
    assertEquals(1, Endereco.findAll().size())
    assertEquals(1, CaracteristicaDoImovel.findAll().size())

    // Executa DELETE
    imovelConsultado.removeFromCaracteristicas(imovelConsultado.caracteristicas.iterator().next())
    imovel.save(flush: true)

    // Verifica cascatas
    assertEquals(1, Localizacao.findAll().size())
    assertEquals(1, Endereco.findAll().size())
    assertEquals(0, CaracteristicaDoImovel.findAll().size())
    assertEquals(0, imovelConsultado.caracteristicas.size())
}
```

ORM: Frameworks

- Erros / Limitações
- Gramática SQL específica para um BD
- Particularidades dos bancos

- Exemplo Hibernate:

```
@GeneratedValue(strategy = GenerationType.AUTO)  
@GeneratedValue(strategy = GenerationType.SEQUENCE)  
// Funciona no MySQL?
```

ORM: Problemas

- Reflexão (entidades com construtor básico...)
- Entidades serializáveis
- Metadados corretos?
- Mapeamento bi-direcional sem dono?
- Representa o design do BD?
 - TablePerClass, SingleTable...

Stored Procedures

- Testes internos nos bancos
- Testes no nível da aplicação
 - Stored Procedure Test
 - Facilidade
 - Menor controle
 - Entrada e Saída

Triggers

- Delay
- Replicação
- BD em comum? Múltiplos usuários?
- Múltiplos desenvolvedores?
- Validação
- Valores gerados

```
ALTER TABLE ENABLE/DISABLE TRIGGER
```

Desempenho

- Querys:
 - Junções (joins) lentas
 - Consultas pesadas
 - Dezenas de milhões de registros
- Conexão: Pool de conexões, # de threads e Timeout
- Para resolver: Coletar o plano de acesso e realizar melhorias de desempenho.

Segurança

- SQLInjection I e II

```
....'; DROP TABLE Tabela; --
```

- Estourar memória:

```
select * from Tabela where x = ' ' or 1 = 1
```

- Erros nos drivers, bases...

Contato

<http://www.agilcoop.org.br>

agilcoop@agilcoop.org.br

helves@ime.usp.br

paulocheque@agilcoop.org.br

