

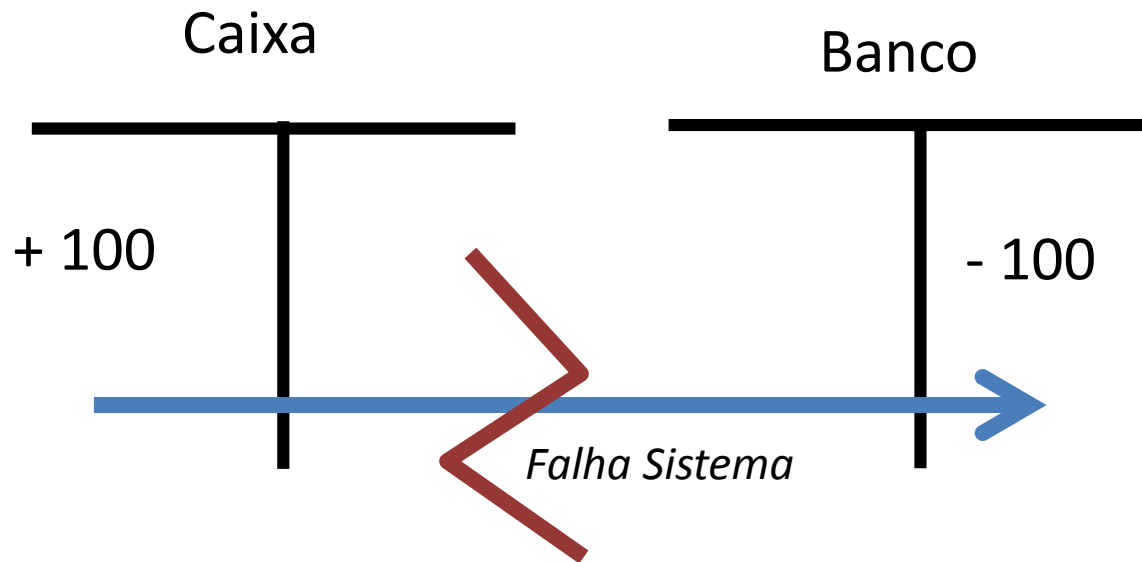
# Transacções e Concorrência

## em Bases de Dados Relacionais

Pedro.Ramos@iscte.pt

## Transacção

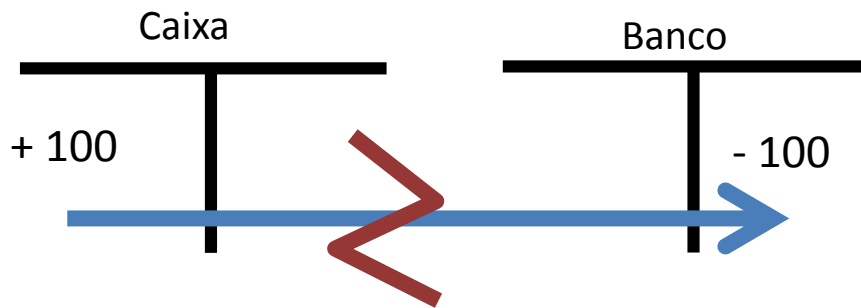
Regra de lançamento contabilístico: débito = crédito



***BD Inconsistente!***

Como manter a regra?

## Transacção



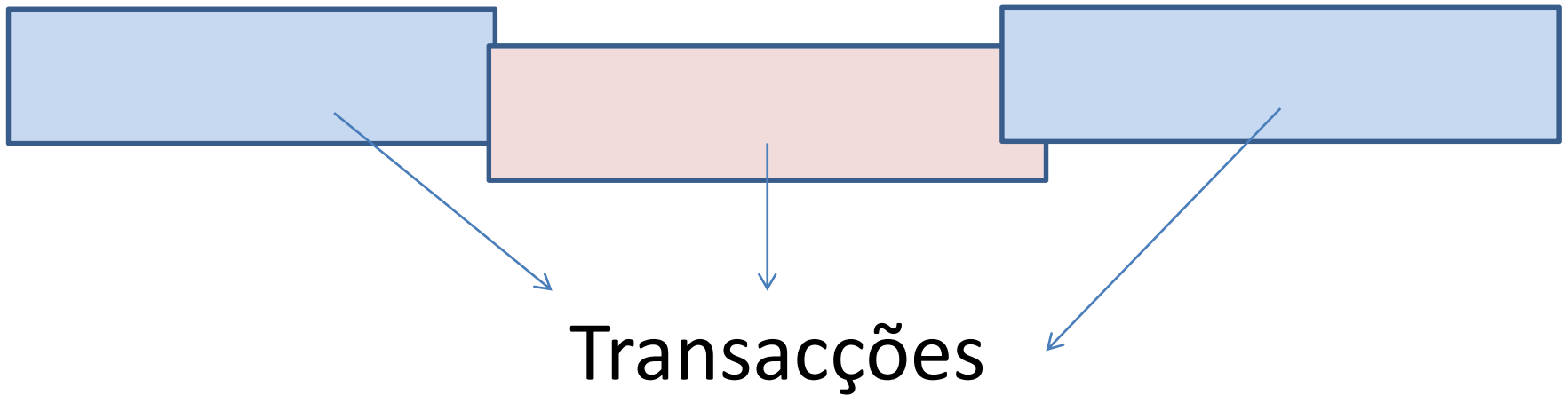
Como manter a regra?

Quando o sistema retomar o controlo, “desfaz” +100.

Mas ..... e se o programa fizer:

+50 Caixa -50 Banco +100 Caixa -100 Banco -200 Caixa +200 Banco

# Transacção



Sequências de acções que não podem ser interrompidas.

# Transacção

**Transacção:** Conjunto delimitado e predefinido de operações que exhibe as seguintes características:

1. **Atomicidade:** grupo indivisível (todas ou nenhuma);
2. **Integridade:** passar de um estado de integridade para outro estado de integridade (bd);
3. **Isolamento:** deve ser executada como se fosse única, ou seja, num ambiente concorrente não pode haver interferências entre transacções. O resultado final é equivalente a uma execução em série (não concorrente).

# Transacção

## Start Transaction

+50 Caixa

-50 Banco

## Commit

+100 Caixa

-100 Banco

## Commit

-200 Caixa

*on error ROLLBACK*

+200 Banco

## End Transaction

**Start Transaction :** inicia uma transacção (termina a anterior).

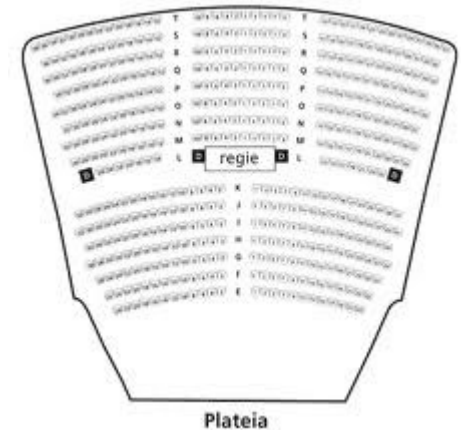
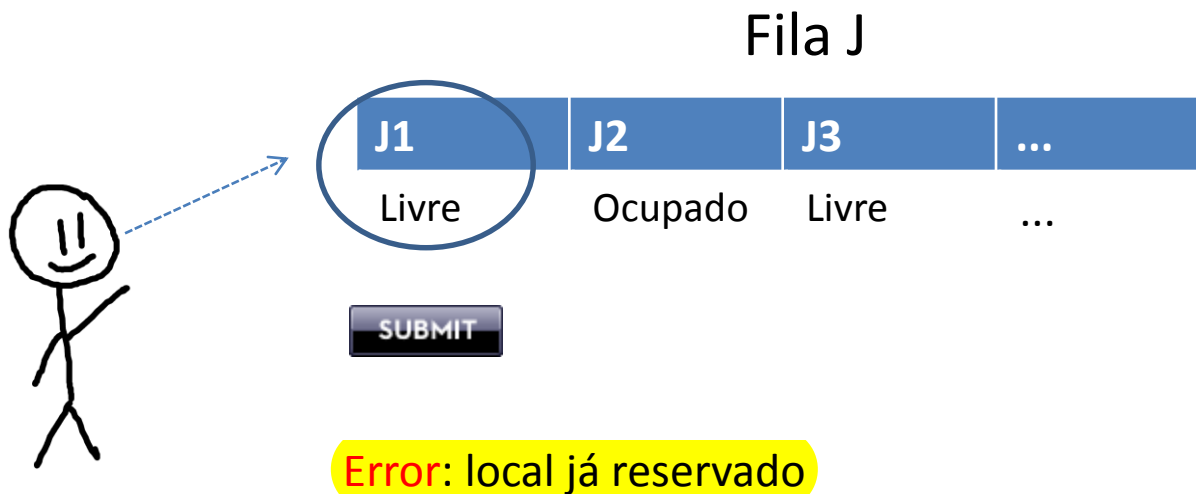
**Commit:** grava definitivamente tudo o que foi feito desde o último commit (ou start transaction).

**Rollback:** desfaz tudo o que foi feito desde o último commit (ou start transaction).

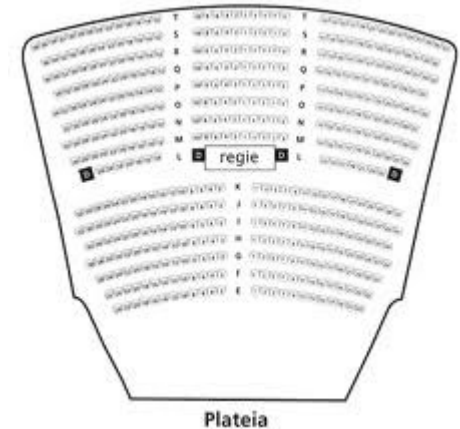
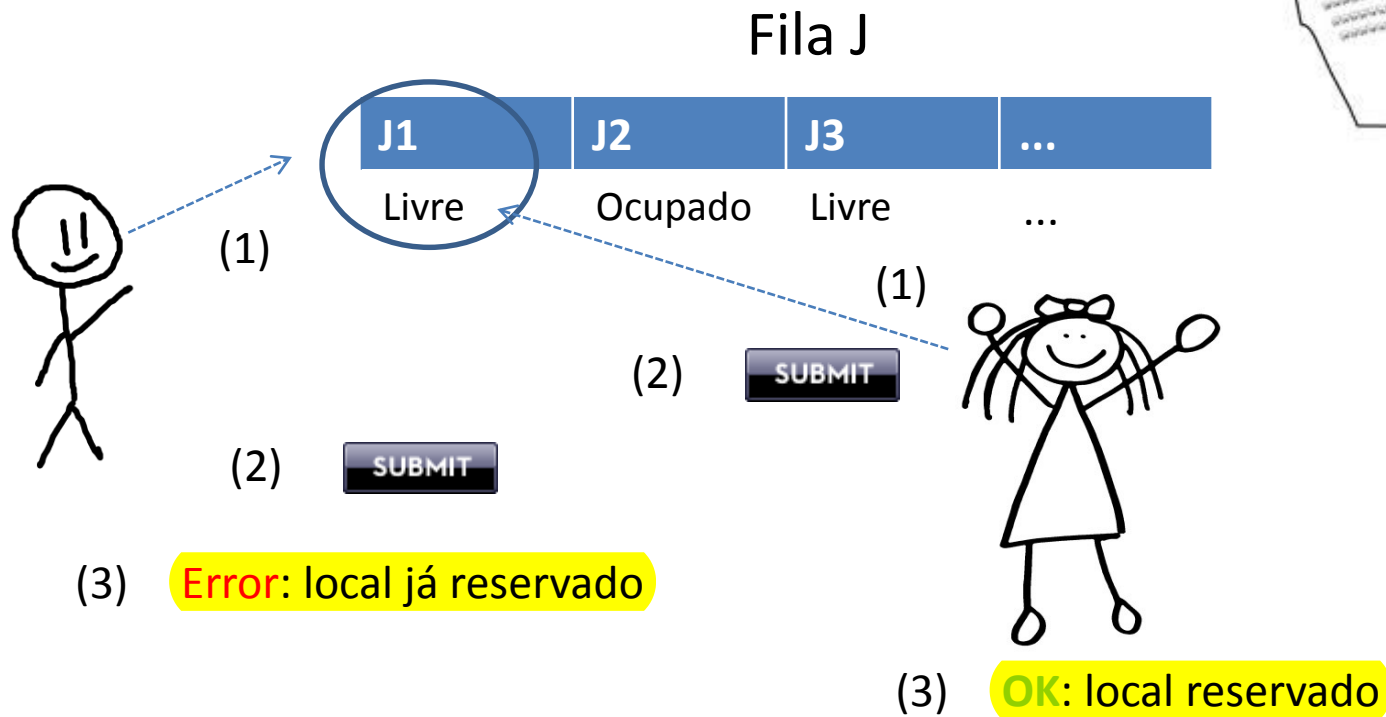
**End Transaction :** finaliza uma transacção (normalmente faz commit , mas pode depender da configuração da bd)

# Concorrência

O problema da concorrência coloca-se em **ambientes multiutilizador**: várias transacções a acederem em simultâneo aos mesmos dados **(por oposição a acederem em série)**.

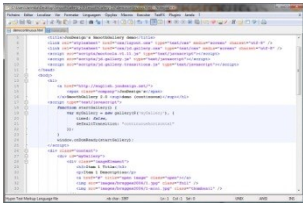


# Concorrência





# Concorrência



(1) Lê Saldo ( $X=1000$ )

(2)  $Y = X/2$

(3) Deposita: Y

(4) Lê Saldo ( $X=1500$ )

(1) Lê Saldo (X=3000)

(2)  $Y = \frac{2}{3} X$

(3) Retira: Y

(4) Lê Saldo ( $X=1000$ )

(5) Escreve X em outra tabela

[illegible]

Os dois programas estão correctos?

# Concorrência



Saldo = 3000

	(1) Lê Saldo ( $X=3000$ )
	(2) $Y=2/3 X$
	(3) Retira: Y
(1) Lê Saldo ( $X=1000$ )	
	(4) Lê Saldo ( $X=1000$ )
(2) $Y=X/2$	
(3) Deposita: Y	
	(5) Escreve X em outra tabela
(4) Lê Saldo ( $X=1500$ )	
(4) Lê Saldo ( $X=3500$ )	

Desfaz tudo

Falha Sistema  
Rollback

# Concorrência

Existem duas formas distintas de resolver problemas , isto é, **serializar os escalonamentos** (um escalonamento concorrente que após o seu término a base de dados fique num estado idêntico ao que teria ficado caso o **escalonamento fosse em série**):

**Métodos Optimistas** – partem do pressuposto que as **interferências são raras**. Deixa ocorrer as transacções até ao fim e depois **verifica se o *commit*** não traz problemas de serialização (utiliza conjuntos ***write set* e *read set*** com todas as actualizações efectuadas). Caso existam problemas faz o ***Rollback*** de tudo.

**Métodos Pessimistas** – utilização de **lockings**, variáveis associadas aos elementos da base de dados que indicam se esses elementos **podem ou não ser acedidos** (para leitura ou/e escrita). Essas variáveis podem **assumir 3 valores**:

**Lock de leitura** – os dados **não podem ser acedidos** nem para **escrita nem para leitura**.

**Lock de escrita** – os dados **não podem ser acedidos** para escrita (**podem para leitura**)

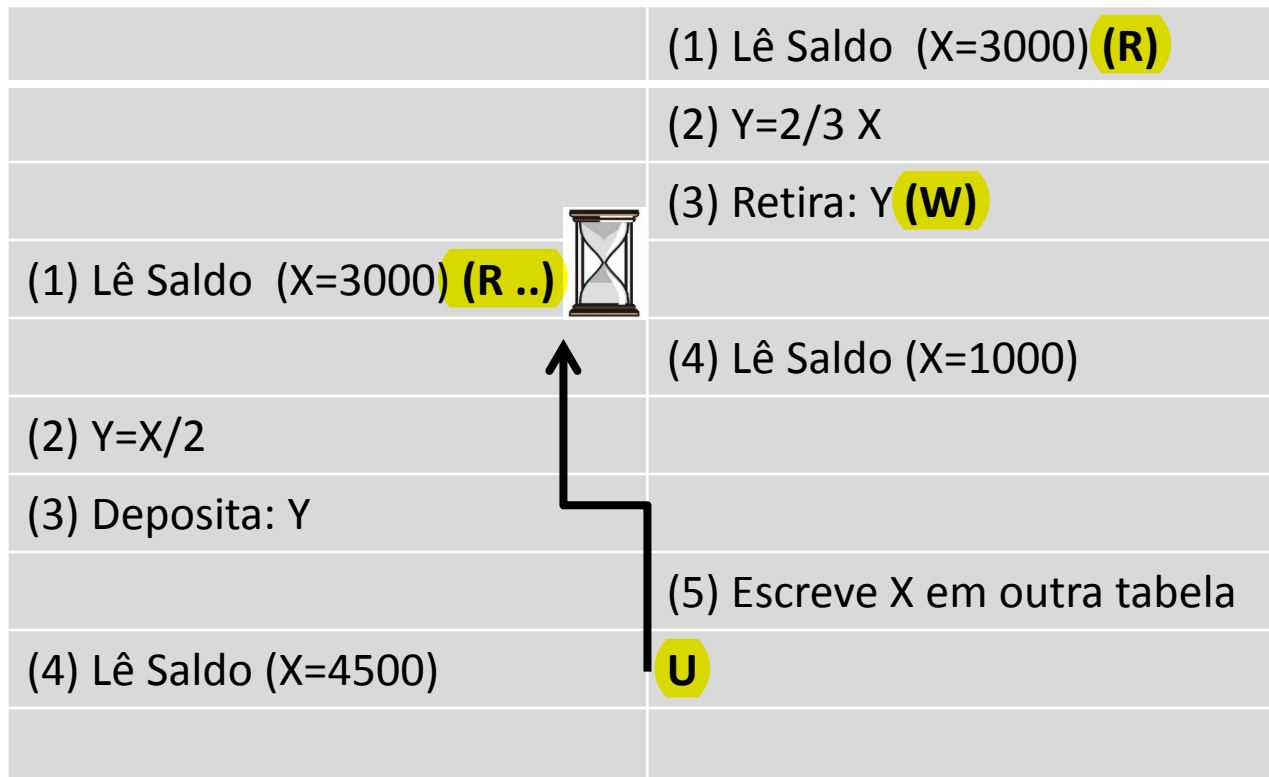
**Unlock** – os dados podem ser **acedidos para leitura e escrita**.

# Concorrência

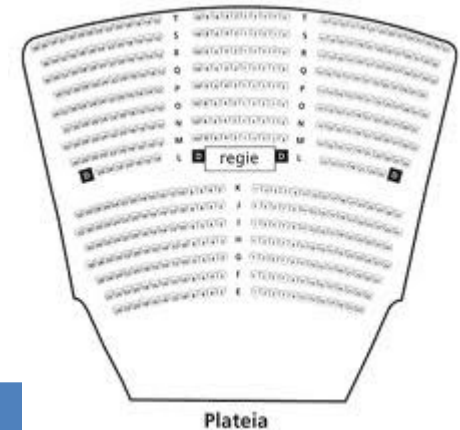
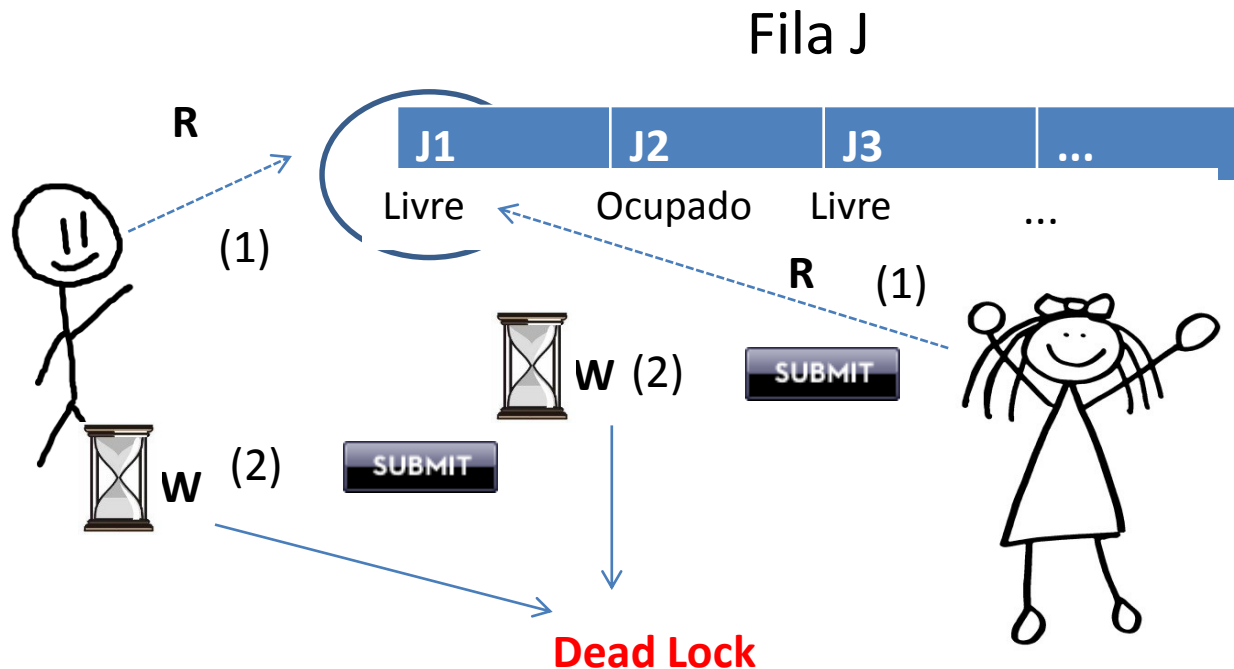
**(W) Lock de leitura** – os dados **não podem ser acedidos** nem para escrita nem para leitura.

**(R) Lock de escrita** – os dados **não podem** ser acedidos para escrita **(podem para leitura)**

**(U) Unlock** – os dados **podem ser acedidos** para leitura e escrita.



# Concorrência (dead lock)




**(W)** os dados não podem ser acedidos nem para escrita nem para leitura.

**(R)** os dados não podem ser acedidos para escrita (podem para leitura)

# Concorrência

**2PL – Two Phase Locking** método de controlo de concorrência que utiliza mecanismos de locking que garantem a serialização das transacções. Resolve todos os problemas excepto dead lock.

Uma transacção satisfaz o 2PL se todos os seus locks antecederem os unlocks. Ou seja, depois do primeiro unlock, não podem haver mais locks.

(1) Lê Saldo (R)	(1) Lê Saldo (R)
(2) $Y=X/2$ (U)	(2) $Y=2/3 X$
(3) Deposita: Y (W) 	(3) Retira: Y (W)
(4) Lê Saldo	(4) Lê Saldo
(U)	(5) Escreve X em outra tabela
	(U)

# Concorrência

**2PL – Two Phase Locking** é muito exigente porque faz com que os recursos (registos) estejam (locked) durante muito tempo (os outros programas têm de ficar à espera e podem ocorrer situações de *time out*).

Também não resolve situações de dead Lock, pelo contrário, potencia-as.

(1) Lê planta (R)	<b>Dead Lock</b>	(1) Lê planta (R)
(2) Escolhe lugar		(2) Escolhe lugar
(3) Reserva Lugar (W)		(3) Reserva Lugar (W)
U		U

(1) Lê planta (R)	(1) Lê planta (R)
(2) Escolhe lugar (U)	(2) Escolhe lugar (U)
(3) Reserva Lugar (W)	(3) Reserva Lugar (W)
U	U

Evita dead lock, mas perigoso.

# Concorrência

Como evitar **dead lock**?

(1) Lê planta <b>(R)</b>	<b>Dead Lock</b>	(1) Lê planta <b>(R)</b>
(2) Escolhe lugar		(2) Escolhe lugar
(3) Reserva Lugar <b>(W)</b>		(3) Reserva Lugar <b>(W)</b>
<b>U</b>		<b>U</b>

Pedir lock para leitura quando apenas se quer ler, quando mais tarde vamos precisar de escrever .

(1) Lê planta <b>(W)</b>	(1) Lê planta <b>(W)</b>
(2) Escolhe lugar	(2) Escolhe lugar
(3) Reserva Lugar	(3) Reserva Lugar
<b>U</b>	<b>U</b>

Evita dead lock, pode  
geralmente **live lock**.



# Concorrência

O que pode um programador fazer?

## 1. Decidir se quer ou não o 2PL

Os SGBD, normalmente, por omissão estão parametrizados para o 2PL. Mas este parâmetro pode ser mudado em tempo real (no meio do código), e no limite, eu posso defini-lo transacção a transacção. A opção não é 2PL ou não: existe uma variável (o nome depende do fabricante) cujo valor varia entre 0 (optimista) e 3 (2PL).

# Concorrência

O que pode um programador fazer?

## 2. Definir as Transacções

Os SGBD, normalmente, por **omissão** estão configurados para um **autocommit**, isto é, cada comando é uma transacção (o lock é sempre libertado, comando a comando). Ou seja, não existe a possibilidade de rollback, logo, por exemplo, se não utilizarmos o 2PL, não existe a possibilidade de “minimizar estragos” (detectar problemas de consistência e recorrer ao rollback para repor uma situação anterior considerada correcta) Em programação pode-se:

....

```
Set autocommit = false;
```

```
Begin transaction
```

```
Select ...
```

```
Update ...
```

```
Select ...
```

```
Commit
```

# Concorrência

O que pode um programador fazer?

## 3. “Gerir” a utilização de locks

Os SGBD atribuem **automaticamente** o **lock em função** do comando SQL:

**Select** – lock de escrita (R);

**Update, Delete, Insert** – lock de leitura (W)

Mas pode-se contornar ...

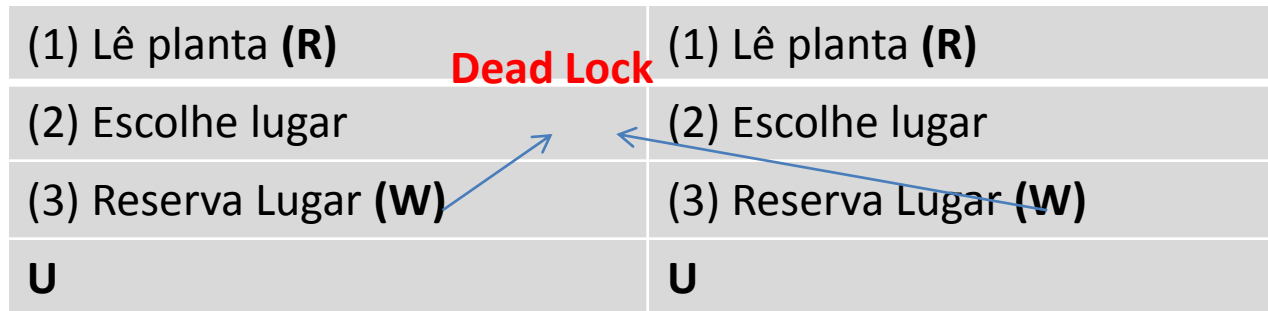
**Select ... For Update ....** – lock de leitura (W), ideal para **evitar dead lock**

**Insert into tabela .... Select ...** - lock de leitura (W), ideal para **chaves incrementais**

# Concorrência

## Exemplo de utilização de locks

(1) Lê planta (R)	<b>Dead Lock</b>	(1) Lê planta (R)
(2) Escolhe lugar		(2) Escolhe lugar
(3) Reserva Lugar (W)		(3) Reserva Lugar (W)
U		U



Pedir lock para leitura quando apenas se quer ler, quando mais tarde vamos precisar de escrever.

(1) Lê planta (W)	 <b>select * from planta for UPDATE</b>
(2) Escolhe lugar	
(3) Reserva Lugar	
U	

# Concorrência

## Exemplo de utilização de locks

### Inserir um novo Cliente

Alternativa 1

```
Select max(idcliente)+1 into_novocliente from cliente;
```

```
Insert into cliente (idcliente) values (novo_cliente) Errado ..porquê?
```

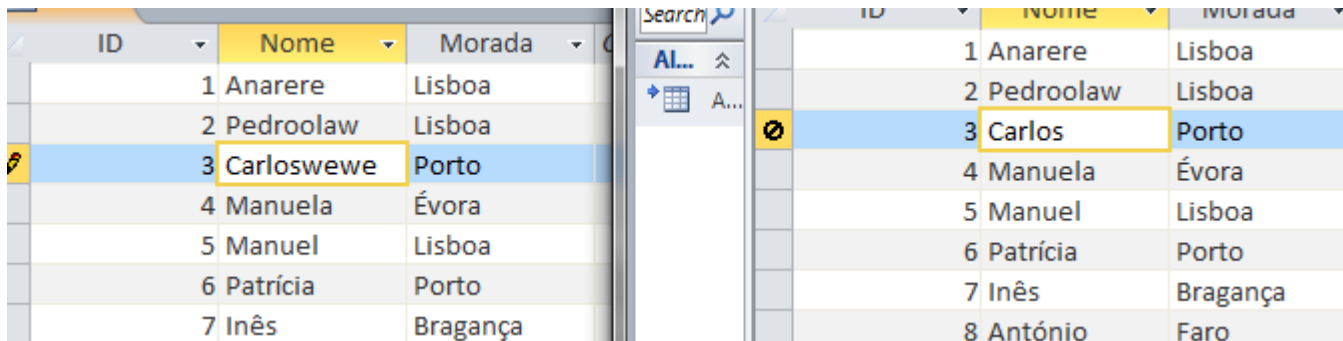
Alternativa 2

```
Insert into cliente (idcliente)
```

```
select max(idcliente)+1 from cliente;
```

# Concorrência

**Lock ao registo? à tabela?**



The image shows two side-by-side screenshots of a database table. The table has three columns: ID, Nome, and Morada. In the left screenshot, the record with ID 3 (Carloswewe, Porto) is highlighted in blue, and a yellow lock icon is visible in the left margin. In the right screenshot, the same record is highlighted, but the lock icon is now a yellow circle with a diagonal line through it, indicating that the lock has been released or is no longer active.

ID	Nome	Morada
1	Anarere	Lisboa
2	Pedroolaw	Lisboa
3	Carloswewe	Porto
4	Manuela	Évora
5	Manuel	Lisboa
6	Patrícia	Porto
7	Inês	Bragança