UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

TECNOLOGIAS DE PROGRAMAÇÃO DE SISTEMAS DE INFORMAÇÃO

TRANSACÇÕES

SISTEMAS GESTORES DE BASES DE DADOS I| Prof. Magno Andrade

Cofinanciado por:

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

UNIÃO EUROPEIA

Fundo Social Europeu

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- Transacções
  - Conjunto de operações de um programa que formam uma unidade lógica de trabalho.
  - Na qual podem ser acedidos e actualizados vários dados.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- Transacções - Exemplo
  - Reserva uma viagem, etc.
  - Transferir dinheiro da conta bancária A para B:

```
1.  read(A)
2.  A := A − 50
3.  write(A)
4.  read(B)
5.  B := B + 50
6.  write(B)
```

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- Transacções
  - Transacção é a visão abstracta que o SGBD tem de um programa de utilizador:
    - Sequência de leituras e escritas.

  - No exemplo anterior

```
read(A)
write(A)
read(B)
write(B)
```

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- Transacções
  - Lista de acções ou instruções:
    - leituras ou escritas sobre objectos da BD.
    - **commit** se a transacção termina com sucesso (por omissão, supõe-se que a transacção termina com sucesso na última instrução).
    - **abort** ou **rollback** se a transacção não termina e todas as suas acções têm que ser anuladas.

UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

PROPRIEDADES ACID

Cofinanciado por:

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

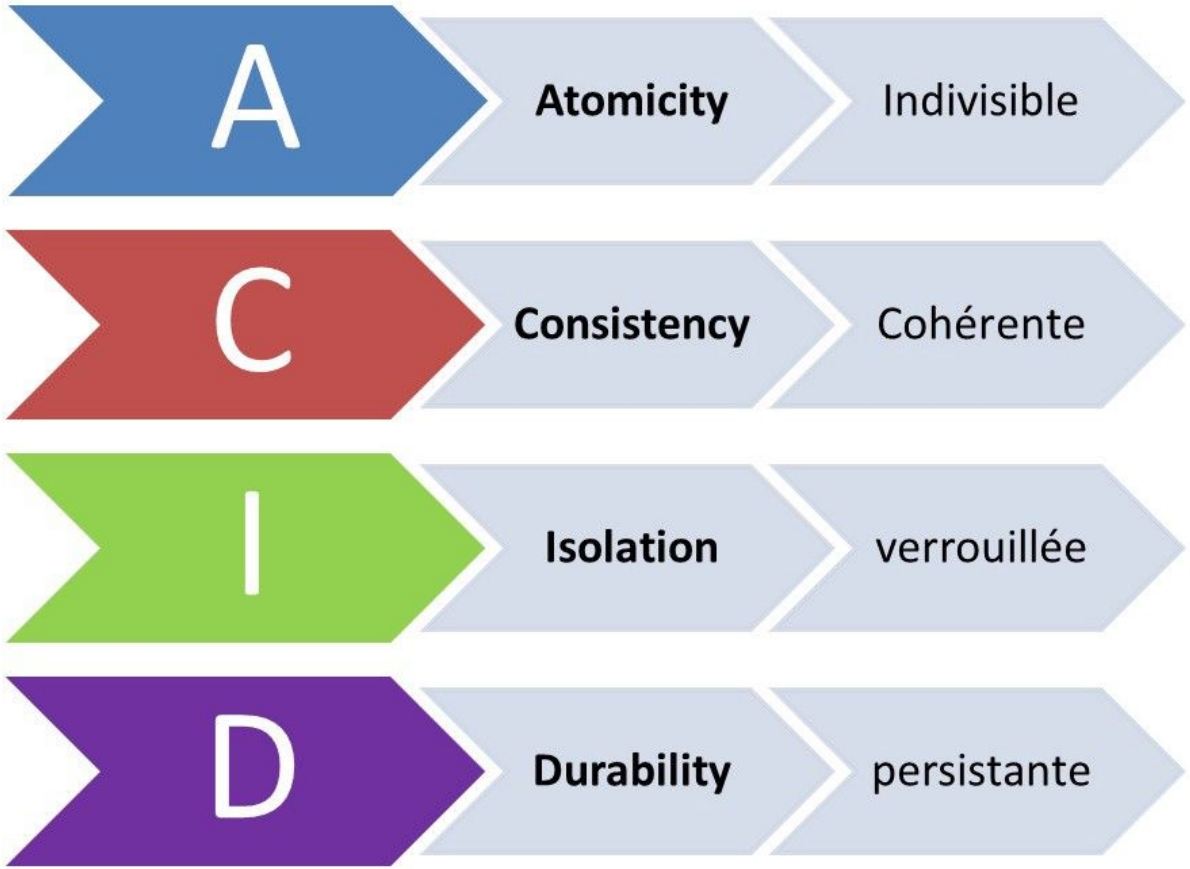
PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Social Europeu

- **Propriedades ACID**
  - As transacções possuem quatro propriedades fundamentais que as caracterizam:
    - Atomicidade
    - Consistência
    - Isolamento
    - Durabilidade

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- **Propriedades ACID**



UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- **Propriedades ACID**
  - Atomicidade
    - se a transacção falhar entre os passos 4–6, os passos 1–3 ficam sem efeito.
  - Consistência (Coerência)
    - a soma A+B tem de ser igual antes e depois.

1	T_i: **read**(A)
2	$A := A - 50$
3	**write**(A)
4	**read**(B)
5	$B := B + 50$
6	**write**(B)

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- **Propriedades ACID**
  - Isolamento
    - nenhuma outra operação deve ler os valores de A e B entre os passos 3 e 6 (verá um valor inconsistente).
  - Durabilidade
    - se a transacção termina com sucesso, as alterações são definitivas, mesmo que o **hw** ou **sw** falhem.

1	T_i: **read**(A)
2	$A := A - 50$
3	**write**(A)
4	**read**(B)
5	$B := B + 50$
6	**write**(B)

UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

ATOMICIDADE

Cofinanciado por:

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Social Europeu

- **Propriedades ACID - Atomicidade**
  - Numa transacção, as alterações ao estado são **atómicas**:
    - ou todas se realizam ou nenhuma se realiza.
  - A transacção deve ser executada na totalidade ou então nenhuma linha o será.
  - Este comportamento “tudo-ou-nada” serve para garantir a consistência em operações que só fazem sentido juntas.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- **Propriedades ACID - Atomicidade**
  - Exemplo:
    - O exemplo clássico que pode ilustrar a necessidade desta propriedade é o processo de transferência de fundos entre duas contas bancárias.
  - Falha de transacções porque:
    - SGBD termina-a devido a anomalia durante a execução.
    - *Crash* de sistema.
    - Transacção encontra situação inesperada (ex: não consegue aceder a disco) e aborta.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- **Propriedades ACID - Atomicidade**
  - Função do sistema
    - Manter informação sobre as alterações efectuadas por cada transacção activa (num ficheiro de *log*).
    - Em caso de aborto, desfazer as alterações das transacções incompletas.

UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

CONSISTÊNCIA

Cofinanciado por:

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Social Europeu

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- **Propriedades ACID - Consistência**
  - Cada transacção deve preservar a coerência nos dados. Para isto ocorrer:
    - Supõe-se que o conjunto das acções da transacção não viola nenhuma das regras de integridade associadas ao estado.
    - Isto requer que a transacção seja um programa correcto.

- **Propriedades ACID - Consistência**
  - Uma transacção deve iniciar a sua execução tendo o sistema um estado conhecido e ao terminar deixá-lo num estado igualmente consistente.
    - Não se podem deixar operações em suspenso para serem tratados por outros blocos/transacções.
    - Caso a transacção aborte, o sistema volta ao estado anterior ao início da execução da transacção e, por este facto, consistente.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- **Propriedades ACID - Consistência**
  - Função do sistema
    - Coerência assegurada por regras de integridade.
    - Não é esperado que o SGBD detecte incoerências nos dados devido a erros na programação.
      - Responsabilidade do Programador/Utilizador

UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

ISOLAMENTO

Cofinanciado por:

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Social Europeu

- **Propriedades ACID - Isolamento**
  - Transacções podem ser executadas concorrentemente.
  - A propriedade de Isolamento garante que:
    - Independente da ordem que as ações sejam executadas em transações concorrentes, o efeito tem que ser idêntico à execução de todas as transações em série.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- **Propriedades ACID - Isolamento**
  - As transacções não devem depender de outras ou influenciar a execução de outras transacções.
  - Cada transacção deve ter a percepção de que está a executar isoladamente no sistema.

- **Propriedades ACID - Isolamento**
  - Quando tal não acontece é vulgar atingir situações de bloqueio mútuo (*deadlocks*).
  - Diferentes transacções estão à espera de outra(s) para poderem prosseguir a sua execução.
  - Situação irreversível que pode conduzir à instabilidade do sistema.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- **Propriedades ACID - Isolamento**
  - Por exemplo:
    - T1 transfere 50€ de A para B.
    - T2 transfere 10€ de A para B.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- **Propriedades ACID - Isolamento**

1	T_1	T_2
	read (A)	
	$A = A - 50$	
	write(A)	
	read(B)	
	$B = B + 50$	
	write(B)	
		read (A)
		$A = A - 10$
		write(A)
		read(B)
		$B = B + 10$
		write(B)

2	T_1	T_2
	read (A)	
	$A = A - 50$	
	write(A)	
		read (A)
		$A = A - 10$
		write(A)
	read(B)	
	$B = B + 50$	
	write(B)	
		read(B)
		$B = B + 10$
		write(B)

- **Propriedades ACID - Isolamento**
  - Duas transacções T1 e T2
    - Transacção T1 cuja execução se sobreponha com a de uma transacção T2.
  - Embora as transações se executem concorrentemente
    - Por exemplo, o escalonamento 2 (slide anterior)
  - Do ponto de vista de T1, esta transacção aparece como tendo sido executada na totalidade antes de T2 ou depois de T2 (execução em **série**).

- **Propriedades ACID - Isolamento**
  - Função do sistema:
    - Garantir que uma transacção apenas “vê” alterações realizadas por outras transacções terminadas com sucesso

UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

DURABILIDADE

Cofinanciado por:

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Social Europeu

- **Propriedades ACID - Durabilidade**
  - Dado uma transacção que foi completada com sucesso (*commit* concluído), o sistema deve garantir:
    - Todas as alterações ao estado são imutáveis, sobrevivendo a qualquer tipo de falta do sistema.
      - Por exemplo, mesmo que haja um *crash* do sistema (antes do registo em disco), o sistema deve garantir que a transacção é refeita.

- **Propriedades ACID - Durabilidade**
  - Após a sua completa execução o objectivo da transacção foi atingido e não deverão existir razões para que algumas das suas instruções sejam anuladas.

    - Qualquer alteração ao estado da base de dados deverão ser efectuadas por outras transacções explicitamente construídas com esse objectivo.

- **Propriedades ACID - Durabilidade**
  - Função do sistema
    - Manter informação sobre alterações efetuadas por cada uma das transacções *commited* (num ficheiro de *log*).
    - Em caso de falha, refazer as alterações que ainda não se encontravam registadas em disco.

UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

RECUPERAÇÃO DE FALHAS

Cofinanciado por:

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Social Europeu

- **Recuperação de falhas**
  - Para anular as acções de uma transacção que abortou (atomicidade), o SGBD mantém um ficheiro de *log* (diário) em que cada escrita é registada.

  - Mecanismo também usado para recuperar de falhas de sistema
    - Todas as transacções activas na altura da falha são abortadas quando o sistema é reposto.

- ***Log***
  - São guardadas as seguintes acções no ***log***:
    - Se ***Ti*** escreve um objecto: valor antigo e valor novo.
      - Registo de *log* tem que ser guardado em disco antes da página mudada o ser.
    - Se ***Ti*** faz ***commit*** ou ***abort***: é escrito registo de log indicando esta acção.
  - Registos de *log* são guardados por transacção para ser fácil anular uma transacção específica.

UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

SQL

Cofinanciado por:

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

UNIÃO EUROPEIA

Fundo Social Europeu

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- **Definição de Transacções em SQL**
  - A linguagem SQL tem elementos para dizer que as consultas fazem parte de uma transacção.
  - Por omissão, o MySQL automaticamente faz ***commit*** permanentemente das alterações na base de dados.
    - Para desactivar:
      - SET autocommit = 0;
      - ou
      - SET autocommit = OFF;

- **Definição de Transacções em SQL**
  - O MySQL para controlar a concorrência utiliza por omissão:
    - bloqueio (*lock*) ao nível da linha/registo e também ao nível da tabela apenas para operações de escrita, mesmo em transacções.

    - é necessário outro tipo de bloqueios (*Internal locking, Isolation levels, Locking Reads*) para assegurar todas as propriedades *ACID*.

- **Exemplo com SQL**
  - Executar uma transacção em SQL
    - **begin** (ou **start**) **transaction**;
    - …
    - **commit**; ou **rollback**;
  - **commit** – torna os resultados permanentes.
  - **rollback** – cancela todas as alterações (*undo*).

- **Exemplo com SQL**
  - Vários sistemas usam *autocommit* por omissão
    - Se ***start transaction*** for omitido
      - cada consulta é uma transacção
      - se houver erros, ***rollback*** automático
      - se não houver erros, ***commit*** automático

- **Exemplo com SQL**
  - Verificar saldos:
    - select balance from account where account_number = 'A-101';
    - select balance from account where account_number = 'A-102';

- **Exemplo com SQL**
  - Transferir 100€ da conta A-101 para a conta A-102:
    - start transaction;
      - update account set balance = balance – 100 where account_number = 'A-101';
      - update account set balance = balance + 100 where account_number = 'A-102';
    - commit;

UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

ANOMALIAS EM CONCORRÊNCIA

Cofinanciado por:

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

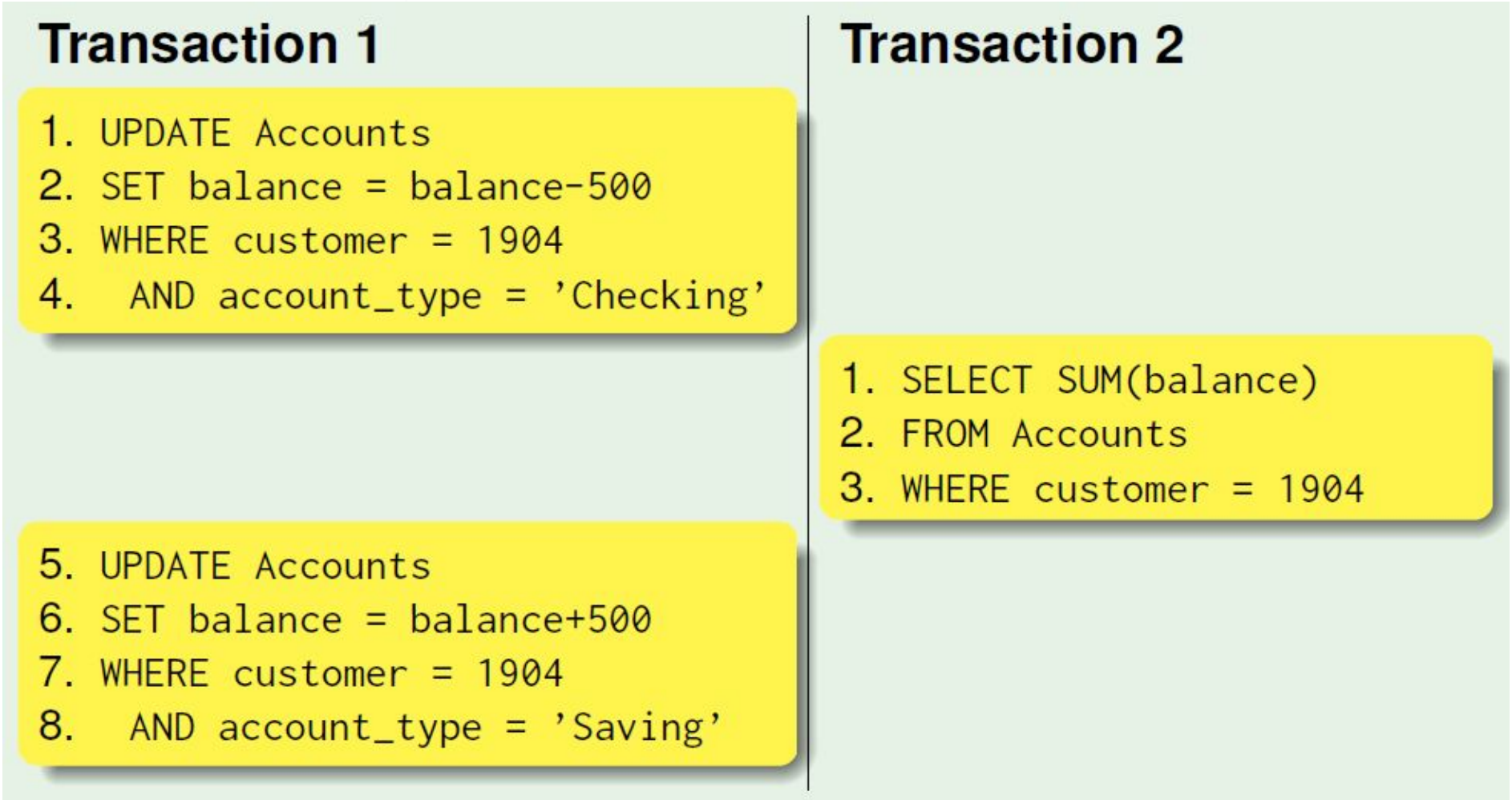
UNIÃO EUROPEIA
Fundo Social Europeu

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS I

- **Anomalias em Concorrência - *Lost Update***
  - Duas pessoas tem cartão de débito para a mesma conta bancária.
    - E se as duas pessoas usarem os cartões ao mesmo tempo?
    - O *update* da segunda pessoa foi perdido.

Primeira pessoa	**Segunda pessoa**	**Estado da base de dados**
bal <- read(account)		12000
	bal <- read(account)	12000
bal <- bal - 1000		12000
	bal <- bal - 2000	12000
	write(account; bal)	10000
write(account; bal)		11000

- **Anomalias em Concorrência - *Inconsistent Read***



- **Anomalias em Concorrência - *Inconsistent Read***

    - A Transacção 2 vê um estado da base de dados temporário e inconsistente.

- **Anomalias em Concorrência - Dirty Read**
  - Duas pessoas tem cartão de débito para a mesma conta bancária.
    - E se as duas pessoas usarem os cartões ao mesmo tempo?
    - A transacção da segunda pessoa leu o saldo modificado antes da transacção da primeira pessoa tenha realizado o *rollback*.

Primeira pessoa	**Segunda pessoa**	**Estado da base de dados**
bal <- read(account)		12000
bal <- bal - 1000		12000
write(account; bal)		11000
	bal <- read(account)	11000
	bal <- bal - 2000	11000
abort		12000
	write(account; bal)	900

UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

SCHEDULER

Cofinanciado por:

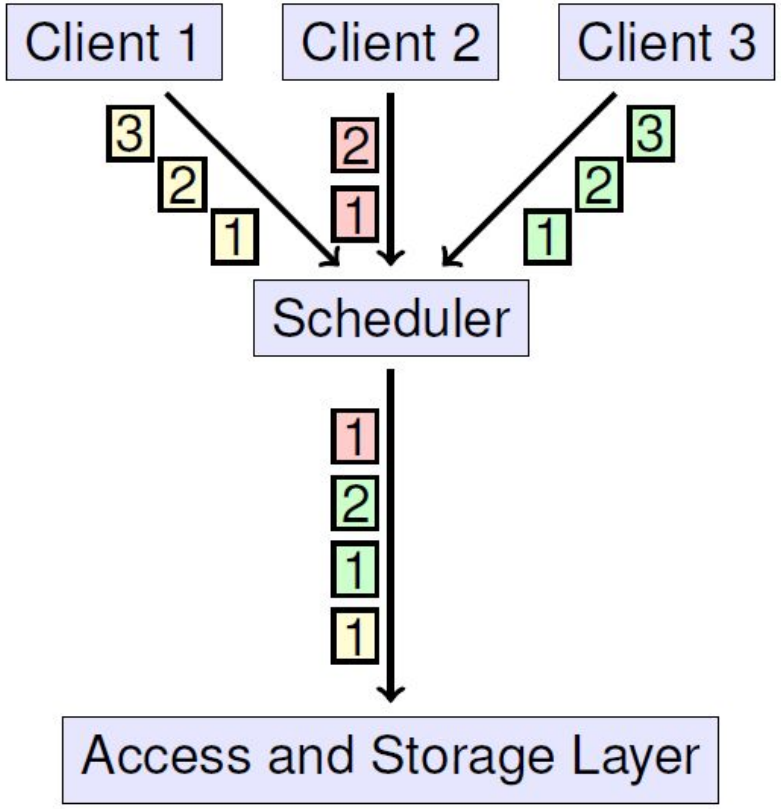
MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Social Europeu

- ***Scheduler***
  - O ***scheduler*** decide a ordem da execução dos acessos concorrentes a base de dados.



- ***Schedules***
  - Um *schedule* é uma lista de acções de um conjunto de transacções.
  - Basicamente é um plano de como executar as transacções.

T_1 : R(V) W(V)
T_2 : R(Y) W(Y)

S_1 :

T_1	R(V)			W(V)
T_2		R(Y)	W(Y)	

S_2 :

T_1	W(V)			R(V)
T_2		R(Y)	W(Y)	

- ***Schedules***
  - Conflitos
    - Duas acções de um schedule estão em conflito se:
      - são de transacções diferentes,
      - envolvendo o mesmo objecto/item a ser acedido,
      - e uma das acções é uma escrita.
    - Vários tipos:
      - *write read (WR)*
      - *read write (RW)*
      - *write write (WW)*

- **Estratégias para assegurar concorrência**
  - Algumas estratégias
    - Pessimista
      - controlo de concorrência baseado em *locks*.
      - controlo de concorrência baseado em *timestamps*.
    - Optimista
      - rastreamento de *read set/write set*.
      - validação antes do *commit* (transacção pode ser abortada).

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Referências**
  - Jörg Endrullis, «Databases».
  - Alberto Sardinha, «Transacções».
  - Helena Galhardas, «Transacções».
  - Hugo Pedro Proença, «Transacções».


UNIVERSIDADE da MADEIRA
CTeSP
CURSOS TÉCNICOS
SUPERIORES PROFISSIONAIS