Transaction (Single Machine)

procedure / function


```
begin_tx
  v1 ← read(checking)
  v2 ← read(saving)
  write(checking - 10)
  write(saving + 10)
end_tx
```

power-down

withdraw(10)

Atomicity : Failure

Consistency : Semantic

Isolation : Concurrency

Durability : Restart

ACID

struct / object .java

{ int / long ... : function

}

A.D. Write-Ahead-Log (WAL)

Redo Log

~~Undo-Log~~

op-log. after-image-log.

encode func.

SQL

```
log-entry ( checking = 90.
            saving = 110
            checksum = md5(y) ) .
```

Reply log.

checkpoint / snapshot.

stop. → flush → checkpoint entry

not stop. flush → entry logs.

requests - logs. snapshot

Concurrency : Isolation.

T_1: Transfer

100 $V_1 \leftarrow R(C)$

100 $V_2 \leftarrow R(S)$

$W(C, 90)$

$W(S, 110)$

T_2: Withdraw.

$100 \leftarrow R(C)$

$W(C, 80)$

$C = 90$ ~~80~~ 2014 Flex Coin

$S = 110$

Ideal Is... Serializability.

$T_1 T_2$ $T_2 T_1$

Equivalent serial

S.S. (Strict-serial----) = Linear---

completion-to-issuing

T_1 $W_1(a)$ $W_3(b)$ C_1 $T_3 T_1 T_2$

T_2 $R_2(a)$ C_2

T_3 $R_3(b)$ C_3

Conflict Serializ— : ordering op. in databases

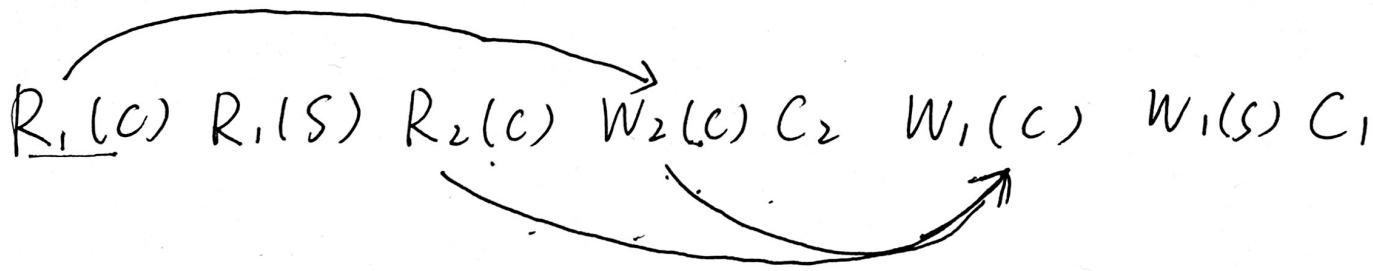
{ R. W.
C. A (bort)

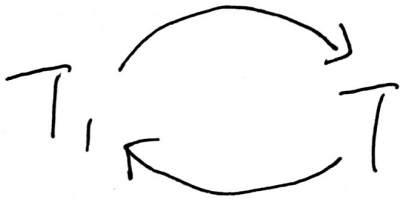
$R_1(C)\ R_2(C)\ R_1(S)\ W_2(C)\ C_2\ W_1(C)\ W_1(S)\ C_1$

Equivalent Schedule - - conflicting op. same order

$R_1(C)\ R_1(S)\ R_2(C)$ - ——— - - - -

Check Serial - - - - : Serialization Graph.





cyclic :

acyclic : serial

topo - - -