

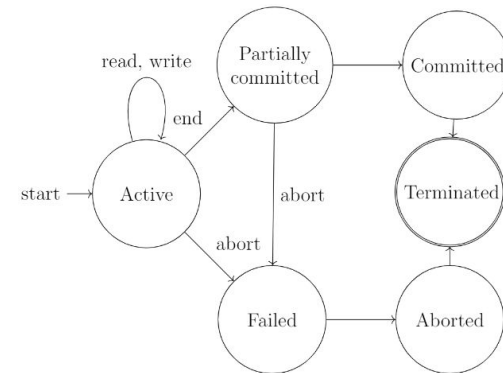
Transacciones

Sivana Hamer - sivana.hamer@ucr.ac.cr
Escuela de Ciencias de la Computación
Licencia: CC BY-NC-SA 4.0

¿Qué problemas existen con las consultas que hemos realizado hasta el momento?

Una transacción es una operación que se ejecuta como una unidad de trabajo lógico

```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```



```
T2 : BEGIN;  
      read(A);  
      A := A + 50;  
      write(A);  
      COMMIT;
```

Las transacciones tienen las siguientes propiedades

A tomicity

C onsistency

I solation

D urability

Atomicity es la propiedad de que se ejecutan todas las operaciones o nada

A = \$1000

B = \$500


```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```

Atomicity es la propiedad de que se ejecutan todas las operaciones o nada

A = \$1000

B = \$500

```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```



Atomicity es la propiedad de que se ejecutan todas las operaciones o nada

A = \$1000

B = \$500

```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```



Atomicity es la propiedad de que se ejecutan todas las operaciones o nada

A = \$1000

B = \$500

```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```



Atomicity es la propiedad de que se ejecutan todas las operaciones o nada

A = \$900

B = \$500

```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```



Atomicity es la propiedad de que se ejecutan todas las operaciones o nada

A = \$900

B = \$500

```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```

Se cae
y se
pierde
\$100



Consistency es que los resultados de la base de datos deben quedar lógicamente consistentes


```
T2 : BEGIN;  
      read(A);  
      A := A + 50;  
      write(A);  
      COMMIT;
```

Resultado correcto de un depósito

Isolation es que ejecutar una transacción concurrentemente debe generar el mismo resultado que ejecución serial

A = \$1000

```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```




Isolation es que ejecutar una transacción concurrentemente debe generar el mismo resultado que ejecución serial

A = \$1000

T1 = \$1000

```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```




Isolation es que ejecutar una transacción concurrentemente debe generar el mismo resultado que ejecución serial

A = \$1000

T1 = \$900

```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```




Isolation es que ejecutar una transacción concurrentemente debe generar el mismo resultado que ejecución serial

A = \$1000

T1 = \$900

```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```

```
T2 : BEGIN;  
      read(A);  
      A := A + 50;  
      write(A);  
      COMMIT;
```



Isolation es que ejecutar una transacción concurrentemente debe generar el mismo resultado que ejecución serial


A = \$1000

T1 = \$900

T2 = \$1000

```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```

```
T2 : BEGIN;  
      read(A);  
      A := A + 50;  
      write(A);  
      COMMIT;
```



Isolation es que ejecutar una transacción concurrentemente debe generar el mismo resultado que ejecución serial


A = \$1000

T1 = \$900

T2 = \$1050

```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```

```
T2 : BEGIN;  
      read(A);  
      A := A + 50;  
      write(A);  
      COMMIT;
```



Isolation es que ejecutar una transacción concurrentemente debe generar el mismo resultado que ejecución serial


A = \$1000

T1 = \$900

T2 = \$1050

```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```

```
T2 : BEGIN;  
      read(A);  
      A := A + 50;  
      write(A);  
      COMMIT;
```



Isolation es que ejecutar una transacción concurrentemente debe generar el mismo resultado que ejecución serial

A = \$900

T1 = \$900

T2 = \$1050

```
T1 : BEGIN;  
      read(A);  
      A := A - 100;  
      write(A);  
      read(B);  
      B := B + 100;  
      write(B);  
      COMMIT;
```

```
T2 : BEGIN;  
      read(A);  
      A := A + 50;  
      write(A);  
      COMMIT;
```

Durability es que no se pierdan datos aunque sucedan errores



Las siguientes son los encargados de las propiedades

A tomicity = Sistema de recuperación

C onsistency = Dev

I solation = Sistema de control de concurrencia

D urability = Sistema de recuperación

Referencias

- R. Elmasri and S. Navathe, Fundamentals of database systems, 7th ed. Pearson, 2016, chapter 20.
- A. Crotty and M. Li. Lecture #15. [Online]. Available: <https://15445.courses.cs.cmu.edu/fall2021/schedule.html>
- A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts, 7th ed. New York, NY: McGraw-Hill, 2020, chapter 17.