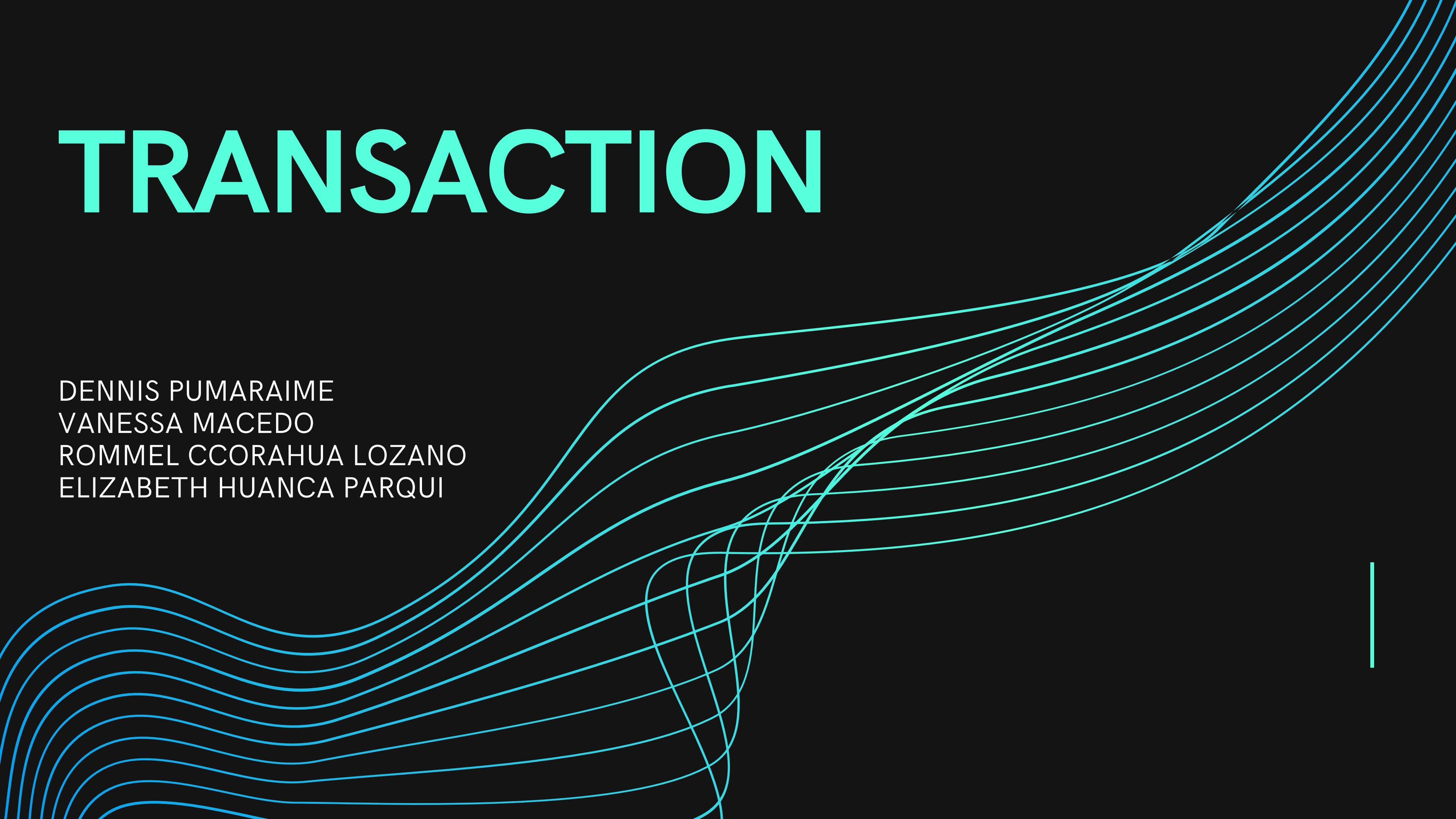


TRANSACTION

DENNIS PUMARAIME
VANESSA MACEDO
ROMMEL CCORAHUA LOZANO
ELIZABETH HUANCA PARQUI



Definicion

QUE ES...



Una transacción es una **unidad de trabajo** compuesta por diversas tareas, cuyo resultado final debe ser que se **ejecuten todas o ninguna** de ellas.



Propiedades ACID

ATOMICIDAD



Las operaciones que componen una transacción deben considerarse como una sola.

CONSISTENCIA

Una operación nunca deberá dejar datos inconsistentes.

AISLAMIENTO

La ejecución de una transacción es independiente una de la otra (simultánea o no).

DURABILIDAD

Los cambios producidos por una transacción deben perdurar (con fallos).

Operaciones

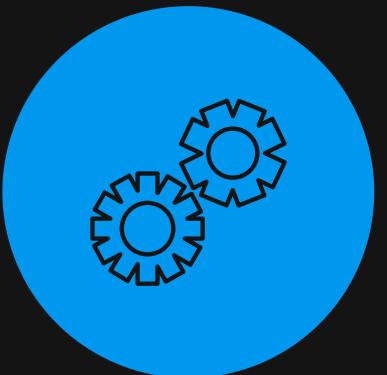
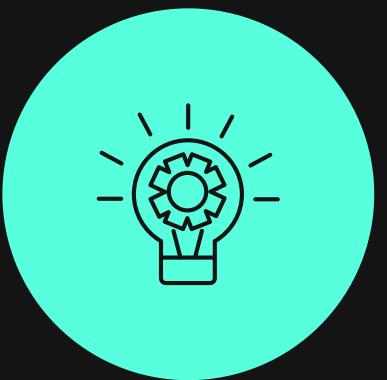


LEER (X)



ESCRIBIR (X)





1. Comprobar que nuestra cuenta existe es válida y está operativa.
2. Comprobar si hay saldo en nuestra cuenta.
3. Comprobar los datos de la cuenta del vendedor (que existe, que tiene posibilidad de recibir dinero, etc....).
4. Retirar el dinero de nuestra cuenta
5. Ingresar el dinero en la cuenta del vendedor.



Bloque

Es la unidad de almacenamiento secundario.



Buffer

Es la unidad de transferencia de información entre el almacenamiento primario y secundario.

Es la unidad de almacenamiento primario

Modelos de Almacenamiento

ALMACENAMIENTO VOLATIL

Falta de energía eléctrica se pierde la información

ALMACENAMIENTO NO VOLATIL

Falta de energía NO se pierde la información

Discos duros, CDs, etc

ALMACENAMIENTO ESTABLE

No importa lo que pase siempre se dispondrá de la información

Múltiples copias

ALMACENAMIENTO SECUNDARIO

No volátil

Almacenamiento Primario □ Es volátil □ RAM

ALMACENAMIENTO PRIMARIO

Es volátil

RAM



NEXT

Transacción Atomicidad y Durabilidad Aislamiento



Una transacción que no finaliza su ejecución con éxito se le dice abortada

Plan de recuperación de la base de datos ->
usa Registro

Beneficio de mantener un registro

Transacción Comprometida y de Compensación

Estado de la

TRANSACCIÓN

ACTIVO

el estado inicial; la transacción permanece en este estado mientras se ejecuta

PARCIALMENTE COMPROMETIDO

Falta de energía NO se pierde la información
Discos duros, CDs, etc

FALLÓ

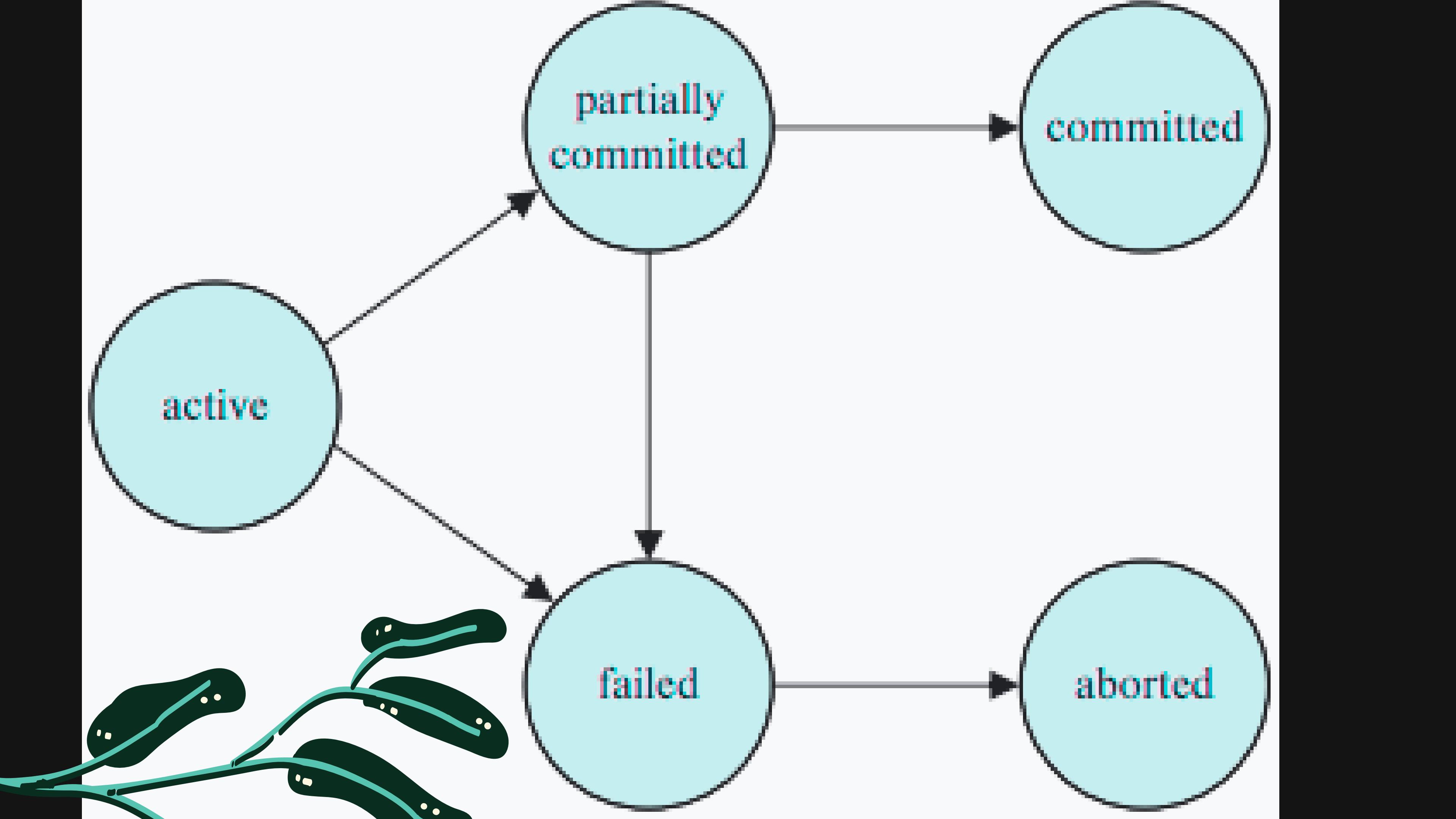
después del descubrimiento de que la ejecución normal ya no puede continuar

ABORTADO

después de que la transacción se haya revertido y la base de datos se haya restaurado a su estado anterior al inicio de la transacción.

COMPROMETIDO

después de completar con éxito



AISLAMIENTO

En bases de datos, el **aislamiento** es una propiedad que define cómo y cuándo los cambios producidos por una operación se hacen visibles para las demás operaciones concurrentes. Aislamiento es una de las 4 propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) aplicables a una base de datos transaccional.



NEXT

Transaction Isolation and Atomicity

Recoverable Schedules

T_6	T_7
$\text{read}(A)$	
$\text{write}(A)$	$\text{read}(A)$
	commit
$\text{read}(B)$	



Transaction Isolation and Atomicity

Cascadeless Schedules

T_8	T_9	T_{10}
$\text{read}(A)$		
$\text{read}(B)$		
$\text{write}(A)$		
	$\text{read}(A)$	
	$\text{write}(A)$	
		$\text{read}(A)$
abort		



Transaction Isolation and Atomicity

Transaction Isolation Levels

SERIALIZABLE

usually ensures serializable execution

REPEATABLE READ

Allows only committed data to be read and further reads. Within a transaction, no other transaction is allowed to update data.

READ COMMITTED

Allows only committed data to be read, but does not require repeatable reads

READ UNCOMMITTED

Allows uncommitted data to be read. It is the lowest isolation level allowed by SQL. usually ensures serializable execution





NEXT

Transacciones como declaraciones SQL

Modelo Simple

```
 $T_i:$  leer(A);  
 $A := A - 50;$   
escribir(A);  
leer(B);  
 $B := B + 50;$   
escribir(B).
```

En SQL:

- Las operaciones de inserción y eliminación son operaciones de escritura
- Los datos específicos (tuplas) a los que se hace referencia pueden estar determinados por un predicado de cláusula where

Por ejemplo

```
select ID, name
from instructor
where salary > 90000;
```

Y si a la vez se **inserta** un nuevo usuario James con salario 100000?

Dependiendo si inserta antes o después el resultado **es distinto**

Entra en **CONFLICTO** =>

Fenómeno
Fantasma

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

La información utilizada para encontrar las tuplas a las que accede la transacción también debe considerarse

Implementacion básica SQL

```
26 # ejemplo de transaccion
27 • START TRANSACTION;
28
29 • SELECT id, nombre FROM instructor WHERE salario > 90000;
30 • INSERT INTO instructor VALUE(99888,"James","Finance",100000.00);
31
32 • rollback; # deshace los cambios de la transaccion
33 • commit; # guarda los cambios de la transaccion
34
```

inicio de la transaccion

cuerpo de instrucciones

Precauciones en las transacciones

- Hay que tener cuidado con los bloqueos , pues provoca que otras tablas tengan que esperar.
- Utilizarlas lo más concisas posibles y solamente donde sea necesario.
- Evitar incluir instrucciones que necesiten mucho tiempo para ejecutarse, como accesar a archivos o a otras bases de datos.
- Los "deathlock" se resuelven haciendo *rollback* a una de las transacciones y permitiendo que la otra se ejecute o haga un *commit* .





good bye

