Transacciones,propiedades ACID y recuperación.

Alex Di Genova

13/06/2023

Informaciones

- Control 2
    - Martes 20 Junio.
    - Sala por informar.
- Control 3
    - Jueves 6 de Julio
        - Sala por informar.
- Control Recuperativo
    - Jueves 13 de Julio

Contenidos

- Repaso (Vistas, Indexación y Optimización)
- Transacciones
  - Propiedades ACID
- Concurrencia y recuperación

Vistas

Ejemplo Join Anidado

- Usos:
  - Para almacenar consultas frecuentes o complejas.
  - Para crear versiones de tablas mas amigables al usuario (tiempo y fechas).

- Crear una vista de un join anidado

```
create view join_album_artista_track as select * from
artist natural join album join track using (albumid)
```

- Como desplegamos los elementos de la lista?

```
select * from join_album_artista_track limit 10;
```

- Como eliminamos la vista?

```
drop view join_album_artista_track;
```

```
[7]  1 %%sql
     2 create view join_album_artista_track as select * from artist natural join album join track using (albumid)
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSources/Chinook_Sqlite.sqlite
Done.
[]
```

```
[8]  1 %%sql
     2 select * from join_album_artista_track limit 10;
```

```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSources/Chinook_Sqlite.sqlite
Done.
```

ArtistId	Name	AlbumId	Title	TrackId	Name:1	MediaTypeId	GenreId	Composer
1	AC/DC	1	For Those About To Rock We Salute You	1	For Those About To Rock (We Salute You)	1	1	Angus Young, Malcolm Young,
2	Accept	2	Balls to the Wall	2	Balls to the Wall	2	1	None
2	Accept	3	Restless and Wild	3	Fast As a Shark	2	1	F. Baltes, S. Kaufman, U. Dirks Hoffman
2	Accept	3	Restless and Wild	4	Restless and Wild	2	1	F. Baltes, R.A. Smith-Diesel, S. Dirkscneider & W. Hoffman
2	Accept	3	Restless and Wild	5	Princess of the Dawn	2	1	Deaffy & R.A. Smith-Diesel
1	AC/DC	1	For Those About To Rock We Salute You	6	Put The Finger On You	1	1	Angus Young, Malcolm Young,
1	AC/DC	1	For Those About To Rock We Salute You	7	Let's Get It Up	1	1	Angus Young, Malcolm Young,
1	AC/DC	1	For Those About To Rock We Salute You	8	Inject The Venom	1	1	Angus Young, Malcolm Young,
1	AC/DC	1	For Those About To Rock We Salute You	9	Snowballed	1	1	Angus Young, Malcolm Young,
1	AC/DC	1	For Those About To Rock We Salute You	10	Evil Walks	1	1	Angus Young, Malcolm Young,

```
#drop view
%%sql
drop view join_album_artista_track;
```

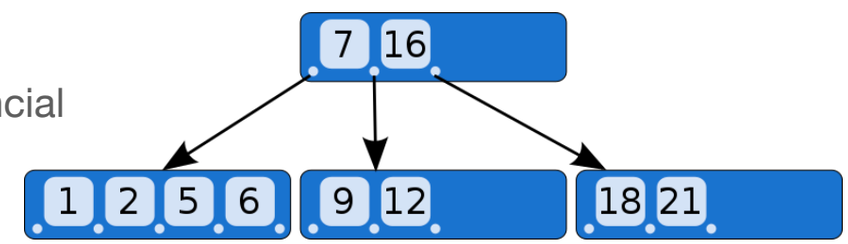
```
 * sqlite:////content/chinook-database/ChinookDatabase/DataSources/Chinook_Sqlite.sqlite
Done.
[]
```

Indices

Optimización de consultas

- Sin indices, el motor de base de datos esta obligado de revisar las tablas completas en cada consulta.
    - Los JOINs son costosos computacionalmente.
    - La idea es encontrar filas sin la necesidad de revisar toda la tabla.
- Cada indice agrega una carga adicional a INSERT, UPDATE y DELETE.
    - Debemos evaluar donde colocar los indices.
- Los indices no se pueden crear en Vistas.
- Como almacena los datos el motor SQL?
- Por defecto, cada tabla es almacenada utilizando una estructura indexada (B-Tree SQLite).
    - A medida que se insertan filas en el B-Tree, las filas se ordenan, organizan y optimizan, de modo que una fila con un ROWID específico y conocido se puede recuperar de manera relativamente directa y rápida.

Permite: busquedas, inserciones, deleciones y accesso secuencial
Time(n)=O(log n)



- Cuando creamos un índice, el sistema de base de datos crea otro B-Tree para almacenar los datos del índice.
- El nuevo B-Tree se ordena y organiza usando la columna o columnas que se especifican en la definición del índice (! ROWID).

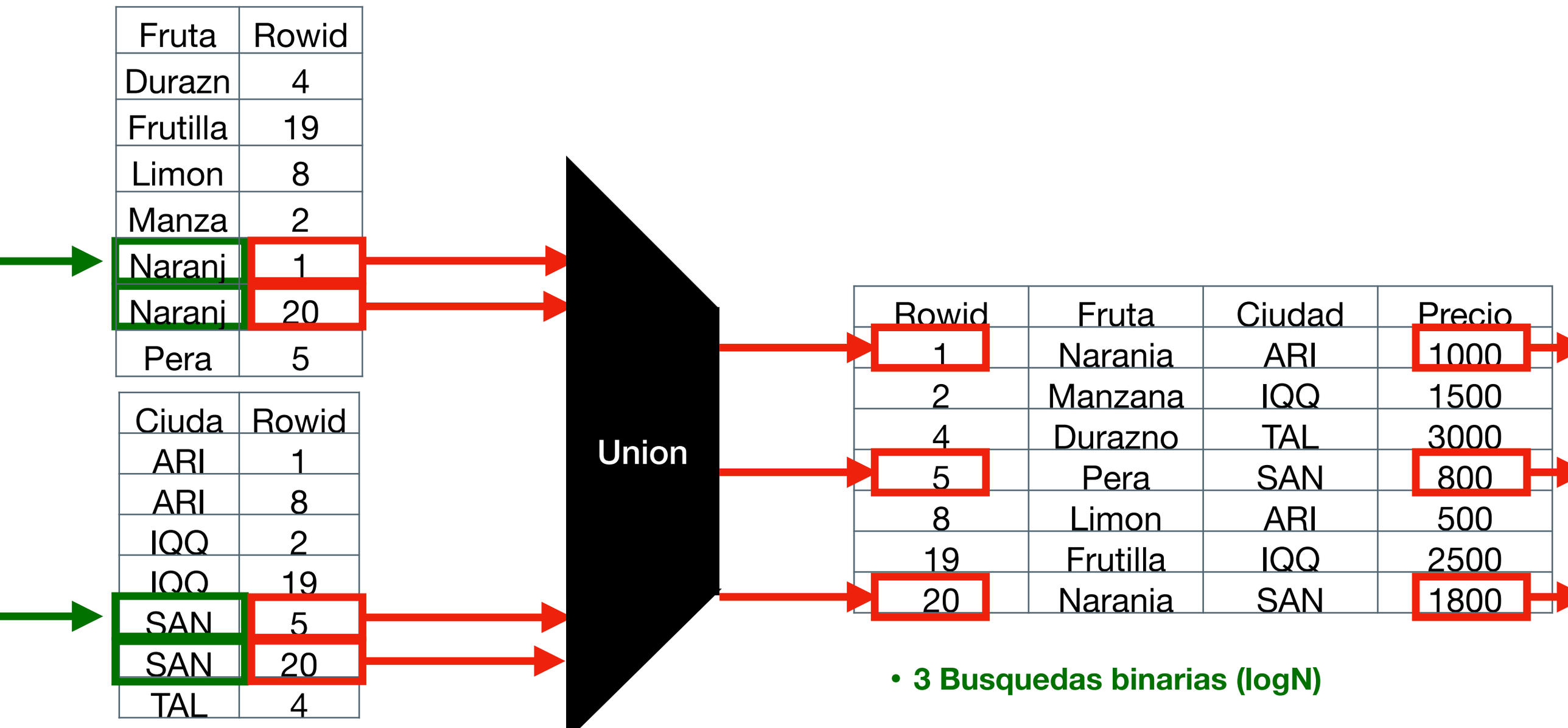
Indices

SQLlite3

- Términos relacionados con OR en la cláusula WHERE

```
SELECT precio FROM VentaFrutas
WHERE fruta='Naranja' OR Ciudad='SAN'
```

- Indices con multiples columnas utiles con AND.

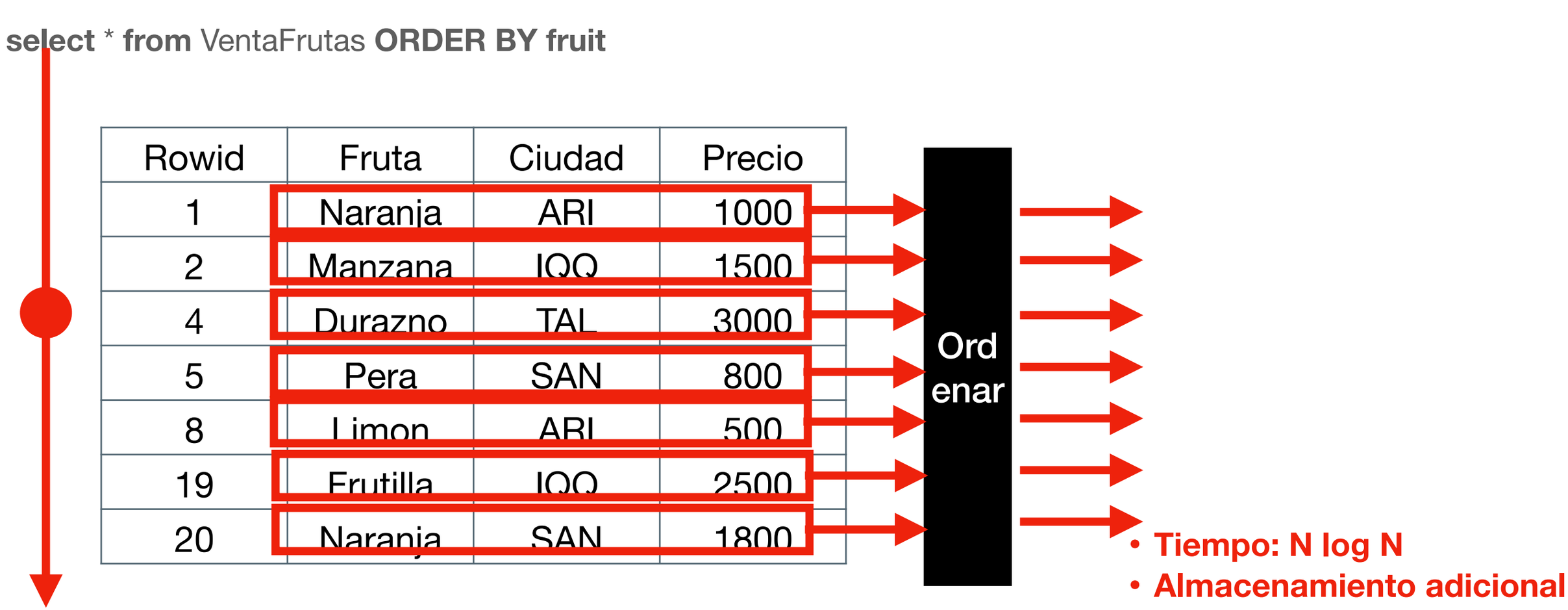


- **3 Busquedas binarias (logN)**

Indices

SQLlite3

- Ordenando
    - Los indices pueden ser usados para acelerar busquedas y ordenamientos.
    - Odenando Tablas sin indices



Optimización SQL

Análisis de la cláusula WHERE

- La cláusula WHERE de una consulta se divide en "términos", donde cada término está separado de los demás por un operador AND.
- Si la cláusula WHERE se compone de restricciones separadas por el operador OR, se considera que toda la cláusula es un único "término" al que se aplica la optimización de la cláusula OR.
- Todos los términos de la cláusula WHERE se analizan para ver si se pueden responder utilizando índices.

```
column = expression
column IS expression
column > expression
column >= expression
column < expression
column <= expression
column IN (expression-list)
column IN (subquery)
column IS NULL
expression = column
expression > column
expression >= column
expression < column
expression <= column
```

Optimización SQL

Análisis de la cláusula WHERE (Ejemplos)

- CREATE INDEX idx_ex1 ON ex1(a,b,c,d,e….,x,y,z) # cobertura
- ... WHERE a=5 AND b IN (1,2,3) AND c IS NULL AND d='hello'
  - a, b, c y d del índice se podrían utilizar, ya que esas cuatro columnas forman un **prefijo del índice y todas están sujetas a restricciones de igualdad.**
- ... WHERE a=5 AND b IN (1,2,3) AND c>12 AND d='hello'
  - [a, b, c] ok d no se puede usar pues sigue a una desigualdad (c > 12).
- ... WHERE b IN (1,2,3) AND c NOT NULL AND d='hello'
  - El índice no se puede usar porque la columna más a la izquierda del índice (columna "a") no está restringida.
- ... WHERE a=5 OR b IN (1,2,3) OR c NOT NULL OR d='hello'
  - El índice no se puede utilizar porque los términos de la cláusula WHERE están conectados por OR en lugar de AND (full table scan).
  - si se agregan tres índices adicionales que contengan las columnas b, c y d como sus columnas más a la izquierda, entonces se podría aplicar la **optimización de la cláusula OR.**

Transacciones

Transacciones

- DDL (Data Definition Language)
    - CREATE TABLE…
- DML (Data Manipulation Language)
    - SELECT …
- TCL (Transaction Control Language)
    - Las transacciones son una parte fundamental de cómo las bases de datos relacionales protegen la integridad y confiabilidad de los datos que contienen.
    - Las transacciones se utilizan automáticamente en todos los comandos DDL y DML.
- Transacción?
    - Una transacción puede involucrar la actualizacion de un solo valor hasta un procedimieto que puede insertar multiples valores en multiples tablas (**update**).
        - Es la unidad basica de cambio en BD (Operaciones parciales no son permitidas)
    - Si dos clientes cambian el mismo registro al mismo tiempo?
        - (Control de concurrencia)
    - Si estamos realizando una transferencia entre bancos y se corta la luz. Cual es el estado de la base de datos?
        - (Durabilidad, recuperación, consistencia)

Transacciones

Realizar una transferencia de dinero

- Como podemos tranferir fondos de manera segura a otra cuenta bancaria.

Pasos	Cliente1	Cliente2
Chequear	X	-
Subtraer	X	-
Agregar	-	X

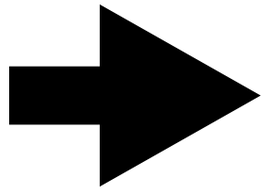
- Todos los pasos deberían tener éxito por completo, dando como resultado que el saldo se transfiera correctamente, o los tres pasos deberían fallar por completo, dando como resultado que ambas cuentas no se modifiquen.

- Cualquier otro resultado, donde un paso tiene éxito y el otro falla, conlleva a a un error.

Transacciones

Realizar una transferencia de dinero

Pasos	Cliente1	Cliente2
Chequear	X	-
Subtraer	X	-
Agregar	-	X

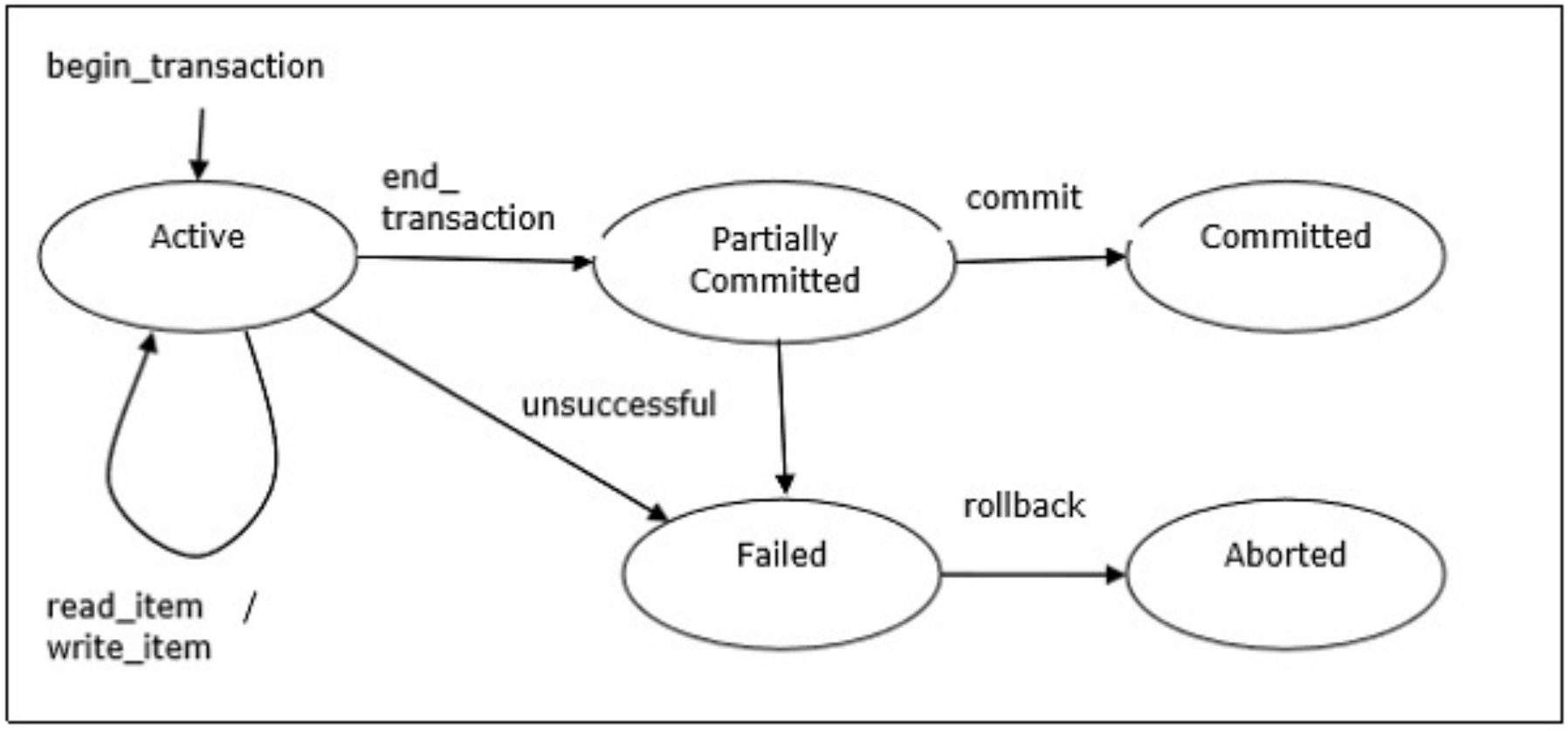


Transacción

- Como garantizar la transacción?
- Ejecutar las transacciones en orden serial:
  - Antes transacción : 1)copiamos DB; 2)cambiamos la copia; 3) si transaccion se ejecuta correctamente? 4) remplazamos la DB original o eliminamos la copia.
- Permitir concurrencia de transacciones independientes (mejor alternativa)
  - Que pasa si cliente1 intenta tranferir 2 veces al mismo tiempo?.
  - Concurrencia: problemas temporales(ok) y permanentes(bad) de consistencia en la DB.

Transaciones

Etapas de una transacción



- BEGIN
  - COMMIT (transacción ok) -> DBMS almacena todos los cambios
  - ABORT (transaccíon falla) -> todos los cambios se deshacen de modo que es como si el txn nunca se ejecutó en absoluto.

Propiedades ACID

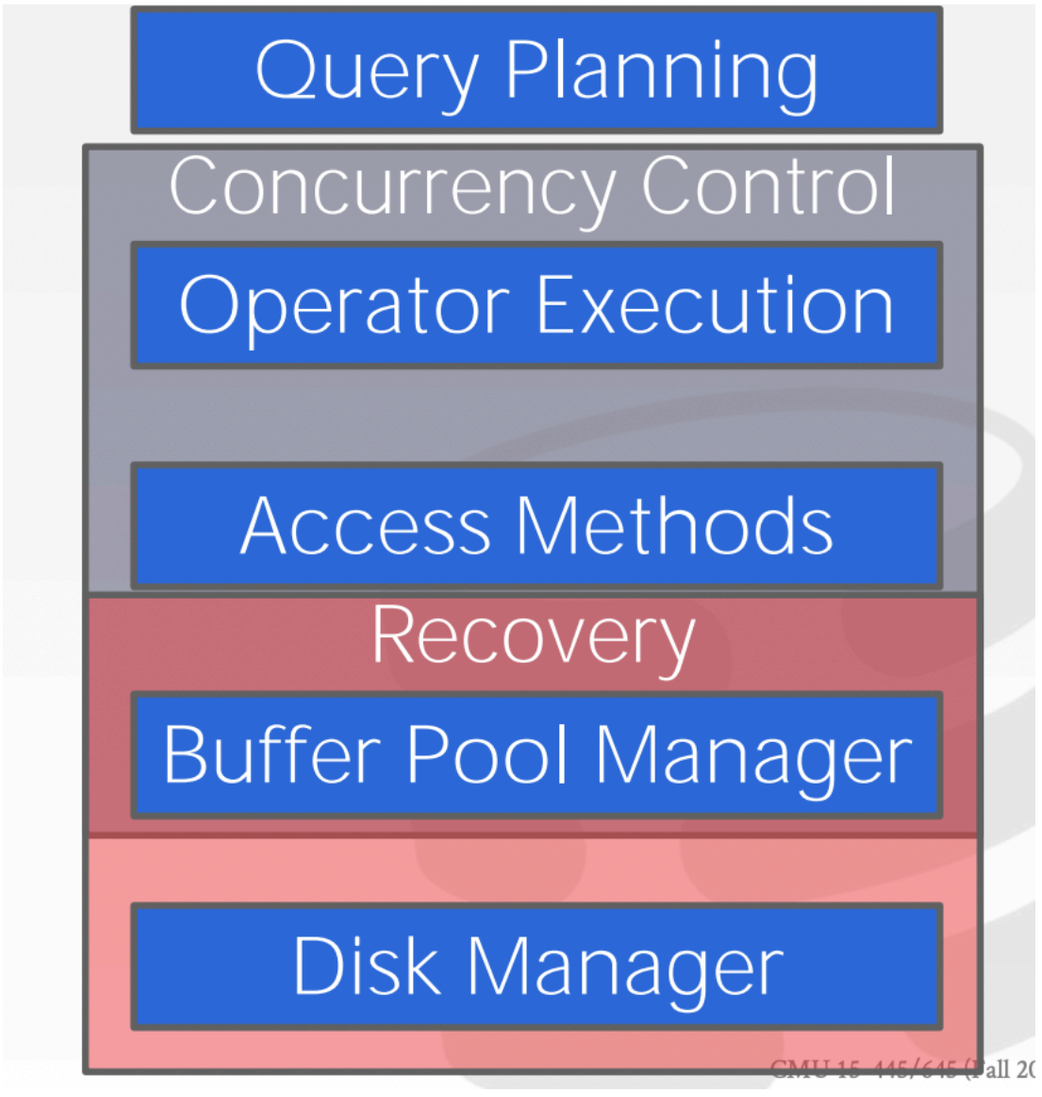
Propiedades ACID

- El estándar para transacciones confiables y sólidas es la prueba ACID.
- ACID significa **Atómica, Consistente, Aislada y Duradera**
- **Atómica:** Una transacción debe ser atómica, en el sentido de que el cambio no se puede dividir en partes más pequeñas. **“Todo o nada”**
- **Consistente**: Suponiendo que una base de datos se inicia en un estado coherente, la aplicación de una transacción debe mantener la base de datos coherente. **“Me parece correcto”**
- **Aislada**: Los cambios no deben ser visibles para ningún otro sistema que acceda a la base de datos, ni deben integrarse en el registro permanente de la base de datos hasta que se confirme toda la transacción. **“Como si estuviera solo”**
- **Duradera:** Una vez que se devuelve un estado de éxito, no debería importar si el proceso se cancela, el sistema pierde energía o el sistema de archivos de la base de datos desaparece; al reiniciar, los cambios confirmados deben estar presentes en la base de datos. **“Sobrevivir a fallas”**
- **Las cuatro propiedades deben cumplirse para garantizar la integridad general de la base de datos.**

Concurrencia y recuperación

Concurrencia y recuperación

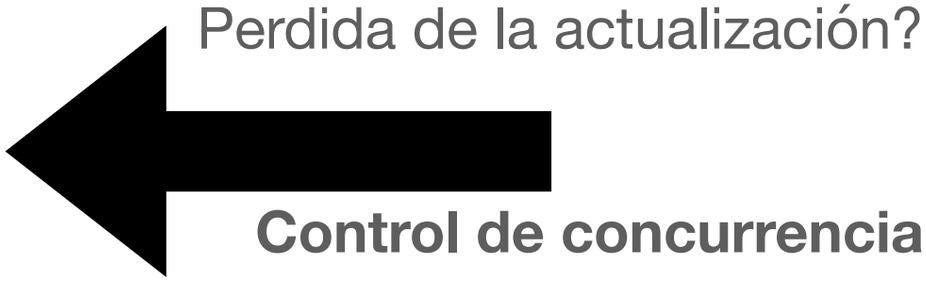
- El control de concurrencia de un DBMS y los componentes de recuperación estan a lo largo del diseño de toda su arquitectura.



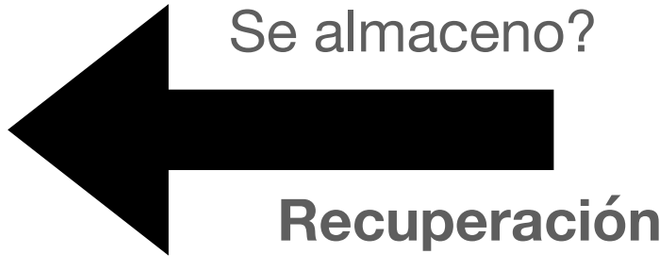
Concurrencia y recuperación

Motivación

- Ambos modificamos el mismo registro en un tabla al mismo tiempo.
  - ¿Cómo evitar y controlar esto?



- Transferi 1M entre banco pero hay un corte de energía.
  - ¿Cuál es el estado correcto de la base de datos?



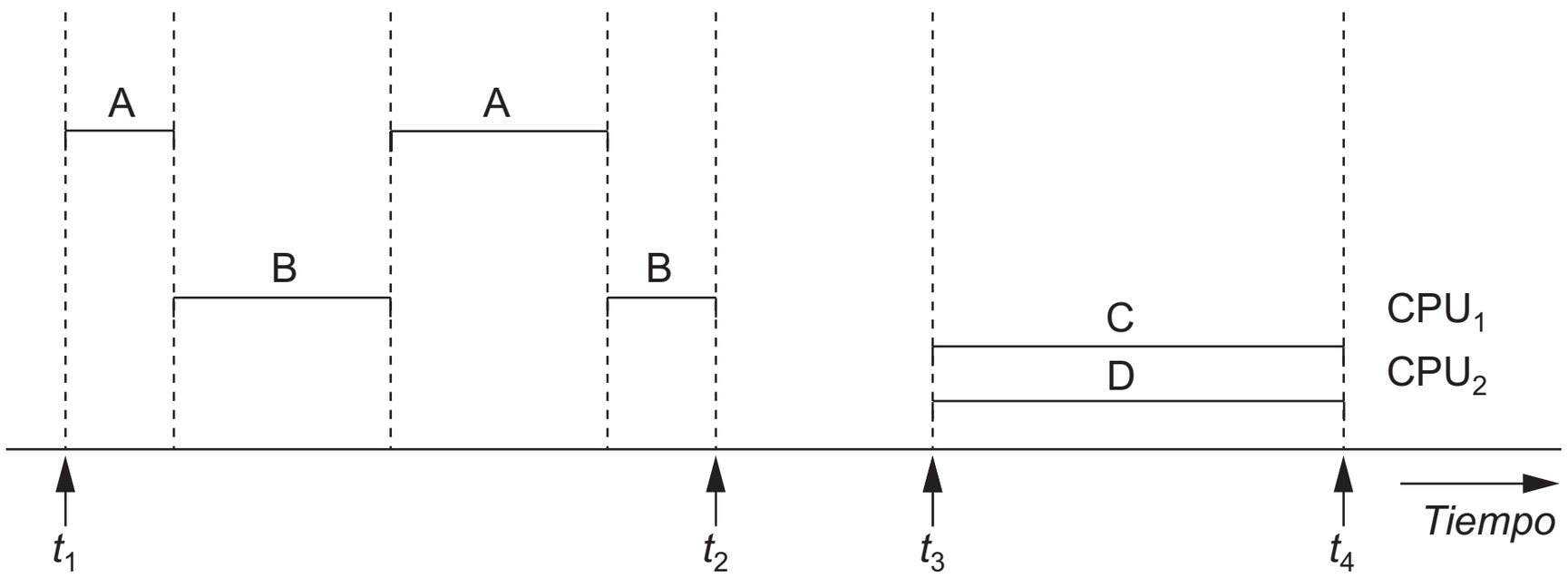
- **Propiedades valiosas de los DBMS.**
- **Basado en concepto de transacciones con propiedades ACID**

Concurrencia

- Los usuarios envían transacciones(tsc), y cada tsc se ejecuta como si fuera unica (aislada).
- El problema de controlar la concurrencia, surge cuando varias tsc enviadas por varios usuarios interfieren entre sí de un modo que produce resultados incorrectos.
- Un protocolo de concurrencia es como el DBMS decide la ejecucion apropiada de las operaciones de multiples tsc.
- Dos categorías de protocolos:
  - → Pesimista: No permitir que surjan problemas (anticipar).
  - → Optimista: Asumir que los conflictos son raros y tratarlos después de que sucedan.

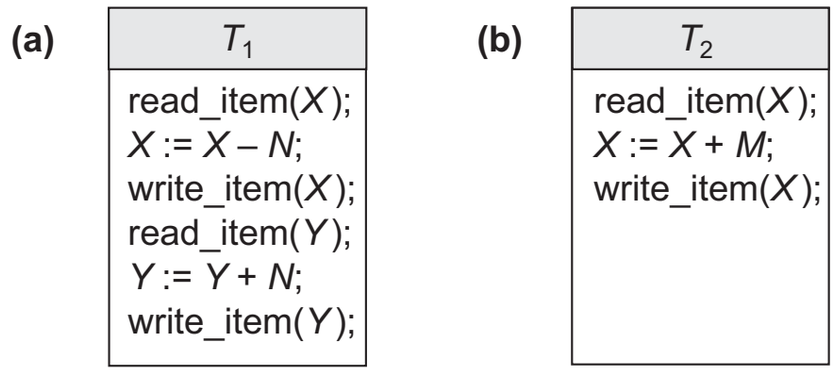
Concurrencia

Interpolado o paralelo



- read_item(X). Lee un elemento de base de datos denominado X y lo almacena en una variable de programa.

- write_item(X). Escribe el valor de la variable de programa X en un elemento de base de datos denominado X.



Concurrencia

Problemas

Time

T_1	T_2
read_item(X);	
$X := X - N$;	
	read_item(X);
	$X := X + M$;
write_item(X);	
read_item(Y);	
	write_item(X);
$Y := Y + N$;	
write_item(Y);	

El elemento X tiene un valor incorrecto porque se ha perdido su actualización realizada por T_1 (sobrescritura).

T_1	T_3
	$sum := 0$;
	read_item(A);
	$sum := sum + A$;
	. . .
read_item(X);	
$X := X - N$;	
write_item(X);	
	read_item(X);
	$sum := sum + X$;
	read_item(Y);
	$sum := sum + Y$;
read_item(Y);	
$Y := Y + N$;	
write_item(Y);	

T_3 lee X después de sustraerse N y lee Y antes de añadirse N; el resultado es erróneo.

Time

T_1	T_2
read_item(X);	
$X := X - N$;	
write_item(X);	
	read_item(X);
	$X := X + M$;
	write_item(X);
read_item(Y);	

La transacción T_1 falla y debe cambiar el valor de X por su valor antiguo;

mientras tanto, T_2 ha leído el valor *temporal* incorrecto de X.

- Problema por pérdida de actualización
- Problema de la actualización temporal (o lectura sucia)
- El problema de la suma incorrecta

Concurrencia

Protocolos

- Bloqueo en dos fases (Pesimista)
    - Determina el orden de serialización de operaciones mientras se ejecutan tsc.
- Ordenamiento por Timestamp (Optimista)
    - Determina el orden de serialización de tsc antes de que se ejecuten.

Consultas?

Consultas o comentarios?

Muchas gracias