



TRANSAKZIO- PROZESATZEAREN KONTZEPTUAK ETA TEORIA



Aurkibidea

- Transakzioaren eta sistemaren kontzeptuak
- Transakzioen exekutatze konkurrentea
- Transakzioak eta berreskuragarritasuna
- Transakzioen propietateak

Transakzioa. Definizioa

- Lan-unitate logikoa da
 - datu-basearen gainean burututako eragiketa-segida (txertatu, ezabatu, aldatu edo eskuratu)
 - bere osaketa programatzaileak erabakitzen du
- Transakzioaren mugak zehazten
 - Hasiera: BEGIN TRANSACTION
 - Amaiera: END TRANSACTION

Transakzioak. Oinarrizko eragiketak

- DBarekin lan egiteko oinarrizko eragiketak:
 - read(X):
DBko X izeneko elementua irakurri eta X izeneko programa-aldagaiari uzten du
 - write(X):
X programa-aldagaiaren balioa DBko X izeneko elementuan idazten du



Granularitatea

- Datu-elementu baten tamaina
- DBko erregistroren bateko eremu bat izan daiteke edo unitate handiagoa izan daiteke, erregistro bat edo disko-bloke oso bat



Transakzioen egikaritzapen konkurrentea

- Erabiltzaile konkurrenteen araberako DBKSa:
 - DBKS erabiltzaile bakarrekoa
 - Aldi berean erabiltzaile bakarra
 - EZ dago konkurrentzia arazorik
 - DBKS erabiltzaile anitzak
 - Erabiltzaile asko momentu oro sistema atzitzen
 - Aginduen exekuzioa → Nola? MULTIPROGRAMAZIOA

Transakzioen egikaritzapen konkurrentea

Hasierako balioak:
 $X = 7, Y = 3$

- Eragiketak exekutatzen diren ordenaren arabera, amaierako emaitza EZ da beti berbera

<u>T1</u>	<u>T2</u>	<u>T1</u>	<u>T2</u>	<u>T1</u>	<u>T2</u>
read(X)	.	.	read(X)	read(X)	.
$X = X * 3$.	.	$X = X + 4$	$X = X * 3$.
write(X)	.	.	write(X)	.	read(X)
read(Y)	.	read(X)	.	write(X)	.
$Y = Y + 2$.	$X = X * 3$.	.	$X = X + 4$
write(Y)	.	write(X)	.	.	write(X)
.	read(X)	read(Y)	.	read(Y)	.
.	$X = X + 4$	$Y = Y + 2$.	$Y = Y + 2$.
.	write(X)	write(Y)	.	write(Y)	.
					7
(a) kasua		(b) kasua		(c) kasua	

Transakzioen egikaritzapen konkurrentea

Hasierako balioak:
 $X = 7, Y = 3$

- Eragiketak exekutatzen diren ordenaren arabera, amaierako emaitza EZ da beti berbera

<u>T1</u>	<u>T2</u>	<u>T1</u>	<u>T2</u>	<u>T1</u>	<u>T2</u>
read(X)	.	.	read(X)	read(X)	.
$X = X * 3$.	.	$X = X + 4$	$X = X * 3$.
write(X)	Bukaeran: $X = 25,$ $Y = 5$.	write(X)	.	read(X)
read(Y)	.	read(X)	.	write(X)	.
$Y = Y + 2$.	$X = X * 3$	Bukaeran: $X = 33,$ $Y = 5$.	$X = X + 4$
write(Y)	.	write(X)	.	.	write(X)
.	read(X)	read(Y)	.	read(Y)	Bukaeran: $X = 11,$ $Y = 5$
.	$X = X + 4$	$Y = Y + 2$.	$Y = Y + 2$.
.	write(X)	write(Y)	.	write(Y)	8
(a) kasua		(b) kasua		(c) kasua	

Transakzioen egikaritzapen konkurrentea

- Transakzioak kolpe bakar batean exekutatuko balira(T1 eta gero T2, edo T2 eta gero T1) emaitza sendoak lortuko genituzke(adib. a eta b kasuak).
- Baina transakzioen eragiketak tartekatzen badira arazoak sortu daitezke (adib. c kasua):
 - Eguneratze galduaren arazoa
 - Behin-behineko aldatzearen arazoa (irakurketa zikina)
 - Irakurketa errepikaezina

Transakzioen egikaritzapen konkurrentea

- Eguneratze galduaren arazoa

T1

```

READ (X)
X = X - N
.
.
WRITE (X)
READ (Y)
.
Y = Y + N
WRITE (Y)

```

T2

```

.
.
READ (X)
X = X + 1
.
.
WRITE (X)

```

X elementuak balio okerra du,
T1-ek egin dion eguneratzea
"galdu" egin baita

Transakzioen egikaritzapen konkurrentea

- Behin-behineko aldatzeak (irakurketa zikinak)

T1

```

READ (X)
X = X - N
WRITE (X)
.
.
.
READ (Y)
... Hutsegitea!

```

T2

```

.
.
.
READ (X)
X = X + 1
WRITE (X)

```

T2-k X-ren behin-behineko balio "okerra" irakurri du

T1-ek huts egiten du eta sistemak X bere balio zaharrera aldatu behar du

Transakzioen egikaritzapen konkurrentea

- Irakurketa errepikaezina

T1

```

•
READ (X)
X = X - N
WRITE (X)
•

```

T2

```

READ (X)
•
•
•
READ (X)

```

T2-k X-aren bi balio desberdin irakurtzen ditu

DBaren sendotasuna

- Bakarka exekutatutako transakzio bakoitzak DBaren sendotasuna gordetzen du
- Baina aldibereko egikaritzapenak ez digu ziurtatzen egoera sendo bat lortuko dugunik
- Ondorioa:
 - Kontrolerako mekanismoa behar dugu, eta mekanismo honek transakzioen exekuzio konkurrenteek sendotasuna mantenduko dutela ziurtatu beharko digu

Berreskuratzearen beharra

- Transakzio guztiek honakoa ziurtatu behar dute:
 - Ondo burutu direla
 - Ez dutela inolako eraginik beste transakzioetan
- Hutsegiteak gerta daitezke

Egunkaria - Log (I)

- log fitxategia
- Hutsegiteetatik berreskuratu ahal izateko egiten den guztia apuntatu
- Gordetzen da:
 - Egunkari-erregistroak:
 - [start_transaction, T]
 - [write, T, X, balio_zaharra, balio_berria]
 - [read, T, X]
 - [commit, T] → Transakzioa ondo bukatu da. Sistema eragileak aldaketak DBn gorde ditzake
 - [rollback, T] → Erroreren bat gertatu da, transakzioa atzera bota da

Egunkaria (II)

TRANSAKZIOA

```
...
insert into Ikaslea values (10, 'Ane', ...)
...
commit
```

(1)

(2)

```
...
[write, T, Ikaslea, null, (10, 'Ane', ...)]
[commit, T]
```

na	izena	...
1	Jon	...
...
10	Ane	...

EGUNKARIA

DATU-BASEA

Egunkaria (III)

- DBaren gainerako aldaketen arrastoa gordetzen du
- DBa berreskuratzeko
 - Desegin: egunkarian atzera joan, commit egin ez duten transakzioen aldaketak DBtik kendu
 - Berregin: commit egin duten transakzioen aldaketak DBan egin

Propietate desiragarriak

- Transakzioek eduki beharko lituzketen ACID/AISI propietateak:
 - **Atomikotasuna** (*atomicity*)
Transakzioa lan-unitate atomikoa da. Bere eragiketa guztiak prozesatzen dira edo bat bera ere ez
 - **Isolamendua** (*isolability*)
Transakzio bat bera bakarrik egongo balitz bezala exekutatu behar da, ezin du beste transakzioen exekuzioan eraginik izan
 - **Sendotasunaren kontserbazioa** (*consistency*)
Transakzioaren egikaritzapen zuzenak DB egoera sendo batetik beste egoera sendo batera eraman behar du
 - **Iraunkortasuna** (*durability*)
Transakzioaren aldaketak hitzartu ondoren, ez dira galdu behar