Gagnasafnsfræði

Samskeiða vinnsla hreyfinga

Hallgrímur H. Gunnarsson

Eiginleikar hreyfinga

Atomicity (A): (all-or-nothing atomicity)

Hreyfing er atómísk á þann hátt að annaðhvort er allt gert eða ekkert gert

Consistency (C):

Hreyfing viðheldur réttleika gagnasafnsins. Ef gagnasafn er í löglegri stöðu þá verður það áfram í löglegri stöðu eftir hreyfingu.

Isolation (I): (before-or-after atomicity)

Ef tvær hreyfingar T_1 og T_2 eru framkvæmdar samtímis þá er niðurstaðan sú sama og að framkvæma þær í röð, þ.e. T_1, T_2 eða T_2, T_1

Durability (D):

Eftir að hreyfing hefur verið staðfest (committed) þá tapast hún ekki.

Aðgerðir í hreyfingum

Hreyfing:

Hreyfing samanstendur af röð aðgerða.

Röðin byrjar á BEGIN og endar á COMMIT eða ABORT/ROLLBACK.

Aðrar aðgerðir eru: lesa hlut (read object) og skrifa hlut (write object).

Til einföldunar hugsum við okkur að gagnasafn samanstandi af einhverskonar *hlutum* (database objects).

Dæmi um hreyfingar

Dæmi:

T_1: BEGIN; A=A+100; B=B-100; COMMIT;
T_2: BEGIN; A=A*1.06; B=B*1.06; COMMIT;

Frá sjónarhóli gagnasafnskerfisins eru þetta aðgerðirnar:

T_1: Read(A); Write(A); Read(B); Write(B);
T_2: Read(A); Write(A); Read(B); Write(B);

Verkröð hreyfinga (e. transaction schedule)

Verkröð:

Verkröð lýsir því í hvaða röð aðgerðir eru framkvæmdar. Aðgerðir sem tilheyra sömu hreyfingu eru alltaf í röð m.t.t. hvors annars.

Dæmi um verkröð:

T_1	T_2
R(A)	
W(A)	
	R(B)
	W(B)
R(C)	
W(C)	

Skilgreiningar

Raðbundin verkröð (e. serial schedule):

Verkröð sem fléttar ekki saman aðgerðum úr ólíkum hreyfingum, þ.e. hreyfingar eru framkvæmdar í e-i röð án þess að skarast.

Ein hreyfing er framkvæmd alveg frá upphafi til enda áður en næsta hreyfing er framkvæmd.

Jafngild verkröð (e. equivalent schedule)

Tvær verkraðanir eru *jafngildar* ef áhrif þeirra á gagnasafnið eru alltaf þau sömu.

Raðbindanleg verkröð (e. serializable schedule)

Verkröð er *raðbindanleg* ef hún jafngildir e-i raðbundinni verkröð.

Dæmi

Hreyfingar:

T_1: Read(A); Write(A); Read(B); Write(B);
T_2: Read(A); Write(A); Read(B); Write(B);

Raðbundin verkröð:

T_1	T_2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(B)

T_1	T_2
	R(A)
	W(A)
	R(B)
	W(B)
R(A)	
W(A)	
R(B)	
W(B)	

Dæmi

Hreyfingar:

T_1: Read(A); Write(A); Read(B); Write(B);
T_2: Read(A); Write(A); Read(B); Write(B);

Raðbindanleg verkröð:

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	
W(B)	
	R(B)
	W(B)
	Commit
Commit	

Vandamál í verkröðum

Skítugur lestur (e. dirty read):

Hreyfing les óstaðfest gögn sem önnur samskeiða hreyfing skrifaði.

Óendurtakanlegur lestur (e. nonrepeatable read):

Hreyfing les tvisvar það sama og fær ólíka niðurstöðu því önnur hreyfing breytir gögnunum og klárast í millitíðinni.

Draugalestur (e. phantom read):

Hreyfing endurframkvæmir fyrirspurn sem sækir lista af röðum sem uppfylla e-ð skilyrði og listinn hefur breyst út af hreyfingu sem var staðfest í millitíðinni.

Einangrunarstig

Stig:

SQL skilgreinir 4 einangrunarstig sem sjást í töflu hér fyrir neðan.

PostgreSQL útfærir í raun bara tvö þeirra: serializable og read committed. Hin stigin skipta minna máli.

Skilgreining á stigum út frá vandamálum:

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read
Read uncommitted	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible
Repeatable read	Not possible	Not possible	Possible
Serializable	Not possible	Not possible	Not possible

Að velja einangrunarstig í SQL

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL <level>
...
COMMIT;
```

þar sem level er:

```
SERIALIZABLE
REPEATABLE READ
READ COMMITTED
READ UNCOMMITTED
```

Default í PostgreSQL er READ COMMITTED.

Útfærsla á samskeiða vinnslu

Einangrunarstig

Hvernig er hægt að tryggja ákveðið einangrunarstig?y

Útfærsla með læsingum

Margir gagnagrunnar nota læsingar til að stýra samskeiða vinnslu (stýra hvaða fléttun er leyfileg) og tryggja einangrun/aðskilnað hreyfinga.

Höfum séð eina læsingarreglu sem tryggir raðbindanleika.

Læsingar eru dýrar og draga úr samskeiða vinnslu.

Útfærsla með myndatöku (e. snapshots)

Nútímaleg leið til að útfæra samskeiða vinnslu hreyfinga er svokallað MVCC kerfi (Multi-Version Concurrency Control), sem aðskilur hreyfingar með því að leyfa margar útgáfur af gögnum. PostgreSQL/Oracle/MySQL nota MVCC.

Læsingar

Lásar

Lás tengist gagnasafnshlut (töflu, línu í töflu, síðu) og stýrir því hver má framkvæma tilteknar aðgerðir á hlutinn.

S-lás (e. shared lock)

S lás er sameiginlegur lás, þá má lesa hlutinn en ekki breyta honum.

Gagnasafnshlutur getur haft marga S-lása samtímis.

X-lás (e. exclusive lock)

X-lás er einkalás, þá má breyta hlutnum.

Gagnasafnshlutur getur bara haft einn X-lás í einu, þ.e. ef hreyfing hefur X-lás á hlut A þá mega aðrar hreyfingar hvorki fá S-lás né X-lás á A.

Læsingarreglur

Strict 2PL

Stíf 2ja fasa læsing (Strict 2-phase locking) er einföld aðferð.

Hún hefur tvær reglur:

1. **Stækkunarfasi**. Ef hreyfing þarf að lesa hlut þá þarf hún S-lás á hann. Ef hreyfing ætlar að breyta hlut þá þarf hún X-lás á hann. Allir lásar eru fengnir áður en einhver er losaður.

2. **Minnkunarfasi**. Hreyfing losar alla lásana sína þegar hún klárast (commit eða rollback)

Stíf 2ja fasa læsing tryggir að allar verkraðanir séu raðbindanlegar (e. serializable)

Samskeiða vinnsla í PostgreSQL

MVCC

Samskeiða vinnsla hreyfinga í PostgreSQL byggir ekki á læsingarreglum heldur notar það svokallað MVCC kerfi (Multi-Version Concurrency Control).

MVCC byggist á því að búa til nýjar útgáfur af gögnum þegar þeim er breytt.

Snapshots

Hægt að taka "mynd" af stöðu gagnagrunnsins á tilteknu augnabliki. Myndin inniheldur þáverandi útgáfu allra staðfestra gagna á tímanum sem myndin var tekin.

Samskeiða vinnsla í PostgreSQL frh.

Framkvæmd hreyfinga

Sérhver hreyfing er framkvæmd í samhengi við tiltekna mynd af stöðu gagnagrunnsins.

Hreyfing getur hvorki séð áhrif frá öðrum óstaðfestum hreyfingum né staðfestingum hreyfingum sem voru staðfestar eftir að hún byrjaði.

Framkvæmd hreyfinga

Þarf reglulega að ruslasafna gömlum röðum sem enginn getur séð.

Læsingar vs. MVCC

MVCC: "Reads do not block writers, writers do not block readers."

Einangrunarstig í MVCC

Einangrunarstig (e. isolation levels)

MVCC í PostgreSQL styður tvö einangrunarstig: SERIALIZABLE og READ COMMITTED.

Raðbindanleg (e. serializable)

Tekin mynd af stöðunni í upphafi hreyfingar. Sú mynd er notuð út alla hreyfinguna.

Lesa staðfest gögn (e. read committed)

Ný mynd er tekin í upphafi hverrar SQL setningar í tiltekinni hreyfingu.

Útfærsla MVCC í PostgreSQL

XID

Sérhver hreyfing fær hlaupandi raðnúmer, s.k. transaction ID eða XID. Hægt að sækja með txid_current().

xmin og xmax

Sérhver lína inniheldur tvo "ósýnilega" dálka með XID númerum: xmin og xmax

xmin er raðnúmer hreyfingarinnar sem bjó til röðina (INSERT eða UPDATE)

xmax er raðnúmer hreyfingarinnar sem úreldi röðina (UPDATE eða DELETE)

Sýnileiki raða í MVCC

Sýnileiki í read committed mode

Röð R er sýnileg í hreyfingu H ef R.xmin er staðfest hreyfing OG (R.xmax er ekki sett EÐA R.xmax er ekki staðfest hreyfing)

Sýnileiki í serializable mode

Röð R er sýnileg í hreyfingu H ef R.xmin er staðfest hreyfing OG R.xmin $<$ H.id OG (R.xmax er ekki sett EÐA R.xmax er óstaðfest hreyfing þegar H byrjaði EÐA (R.xmax er staðfest OG R.xmax $>$ H.id))

Samskeiða skrif í MVCC

Tvær samskeiða hreyfingar geta ekki skrifað (UPDATE, DELETE, SELECT FOR UPDATE) sömu röðina. Ekkert vandamál ef þær skrifa í raðir sem skarast ekki.

Ferli ef hreyfing reynir að skrifa í röð sem önnur hreyfing hefur þegar skrifað:

1. Ef önnur hreyfing er þegar í gangi, þá er beðið þar til hún klárast (commit/abort)
2. Ef það var hætt við hana, þá höldum við áfram (notum upphaflegu útgáfuna)
3. Ef röðin var skrifuð:
   (a) Í SERIALIZABLE: abort with "can't serialize error"
   (b) Í READ-COMMITTED: höldum áfram, notum nýju útgáfuna (nema þetta hafi veirð SELECT FOR UPDATE og röðin uppfyllir ekki lengur WHERE skilyrðið)

Dæmi

SQL

UPDATE webpages SET hits=hits+1 WHERE url='..';

Tvær samskeiða hreyfingar

T_1	T_2
les röð, sér hits=531	
	les röð, sér hits=531
reiknar hits+1=532	
	reiknar hits+1=532
skrifar hits=532	
	skrifar hits=532
commit	
	commit

Dæmi frh.

Raðbindanleg einangrun

T_1	T_2
les röð, sér hits=531	
	les röð, sér hits=531
reiknar hits+1=532	
	reiknar hits+1=532
skrifar hits=532	
	skrifar hits=532
commit	
	error! can't serialize!

Dæmi frh.

Lesa-staðfest einangrun

T_1	T_2
les röð, sér hits=531	
	les röð, sér hits=531
reiknar hits+1=532	
	reiknar hits+1=532
skrifar hits=532	
	reynir að skrifa, þarf að bíða
commit	
	les röð aftur, hits=532
	reiknar hits+1=533
	skrifar hits=533
	commit