Arquitectura de bases de dades

PAC 2: Control de concurrència, gestor de dades i optimització en bases de dades distribuïdes

Pregunta 1 (2,5 punts)

Enunciat

Per tal de realitzar aquest exercici, cal llegir l'article "NewSQL database systems are failing to guarantee consistency, and I blame Spanner" de Daniel Abadi (http://dbmsmusings.blogspot.com/2018/09/newsql-database-systems-are-failing-to.html). Es demana que contesteu a les preguntes següents:

Nota: la resposta a aquest exercici no ha de superar l'extensió de dues pàgines

a) Segons l'autor, quines de les garanties del teorema CAP satisfan els sistemes de bases de dades distribuïdes NoSQL i NewSQL? En què es basa aquest teorema?
b) Quines tres raons dóna l'autor per a que els sistemes moderns siguin CP (de CAP) en lloc d'AP i per què?
c) Com es pot garantir la consistència en els sistemes distribuïts segons l'autor? Quines dues categories identifica als sistemes NewSQL?
d) Quines tècniques utilitza Spanner per garantir la consistència?

Criteris d'avaluació

Els criteris d'avaluació que s'aplicaran en la correcció d'aquesta pregunta són els següents:

- Les preguntes no contestades no penalitzen
- Es valorarà la qualitat de la resposta
- Es valorarà la concreció en l'argumentació de la resposta
- Totes les preguntes tenen el mateix pes.
- Es valorarà no superar l'extensió màxima recomanada.

Solució

a) Segons l'autor, quines de les garanties del teorema CAP satisfan els sistemes de bases de dades distribuïdes NoSQL i NewSQL? En què es basa aquest teorema?

El teorema CAP afirma que un sistema de base de dades distribuït no pot garantir alhora les tres característiques següents:

- Consistència (***C****onsistency*): la consistència ens garanteix que una lectura ens retornarà l'escriptura més recent d'un registre donat. El que això implica és que sempre que es faci alguna modificació a una dada, aquest canvi s'ha de reflectir en tots els nodes de la base de dades. Això garanteix que sempre que s'accedeixi a la informació qualsevol dels nodes pot respondre i la informació sempre serà la mateixa.
- Disponibilitat (***A****vailability*): un node en funcionament ens ha de retornar una resposta raonable en un període raonable de temps (ni error ni *timeout*).
- Tolerància al particionament (***P****artition tolerance*): el sistema ens ha de seguir funcionant encara que alguns nodes no estiguin disponibles ja que la informació és consistent en tots els nodes.

El teorema diu que és impossible garantir tant la consistència com la disponibilitat en el cas d'una partició de xarxa. D'aquesta manera els arquitectes de sistemes moderns de bases de dades escalables, es van dividir en dos camps que es poden identificar a grans trets com: els que prioritzaven la disponibilitat (NoSQL) i els que prioritzaven la consistència (NewSQL).

b) Quines tres raons dóna l'autor per a que els sistemes moderns siguin CP (de CAP) en lloc d'AP i per què?

(1) Els sistemes que no garanteixen la consistència donen com a resultat un codi d'aplicació complex, costós i sovint amb errors. A banda calen perfils de desenvolupadors molt especialitzats per generar aquest codi.

(2) La reducció de la disponibilitat causada per la garantia de consistència és mínima i amb prou feines perceptible en molts desplegaments.

(3) El teorema CAP és fonamentalment asimètric. Els sistemes CP poden garantir la consistència mentre que els sistemes AP no garanteixen la disponibilitat (cap sistema pot garantir el 100% de disponibilitat). Per tant, només un costat del teorema CAP obre la porta a qualsevol garantia útil.

L'autor considera que aquests tres punts han fet que els sistemes distribuïts que garanteixen la consistència hagin ressorgit per davant dels que garantien la disponibilitat. Per a molts desenvolupadors és més segur desenvolupar per sobre de sistemes CP.

c) Com es pot garantir la consistència en els sistemes distribuïts segons l'autor? Quines dues categories identifica als sistemes NewSQL?

Hi ha moltes maneres de garantir la consistència en els sistemes distribuïts. El mecanisme més popular per garantir la consistència amb la mínima pèrdua de

disponibilitat és l'ús de protocols de consens que garanteixen la consistència a través de múltiples rèpliques de les dades. De manera simplificada aquests protocols funcionen mitjançant un mecanisme de votació per majoria. Qualsevol canvi en les dades requereix que la majoria de les rèpliques hi estiguin d'acord. Això permet que una minoria de rèpliques estiguin inactives o no disponibles, però que el sistema pugui seguir llegint o escrivint dades.

Molts dels sistemes NewSQL utilitzen protocols de consens per aconseguir la consistència. De tota manera difereixen en com utilitzen aquests protocols de consens. L'autor estableix dues categories d'aquests sistemes:

- La que utilitza un únic protocol de consens per base de dades. Cada transacció s'executa en el mateix protocol global. Ex. Calvin i Fauna DB.

- La que divideix les dades en fragments i aplica un protocol de consens per a cada fragment de dades. Ex. Spanner, CockroachDB i YugaByte.

d) Quines tècniques utilitza Spanner per garantir la consistència?

Spanner va fer front a aquest inconvenient mitjançant la seva API *True Time* que consisteix en que totes les transaccions reben una marca de temps que es basa en l'hora actual. D'aquesta manera es pot determinar donades dues transaccions diferents quina s'executa "abans" i quina "després". El fet però, que hi hagi diferents servidors, pots fer que hi hagi petites diferències de temps entre els rellotges dels servidors. Per aquest motiu, Spanner estableix una finestra d'"incertesa" que es basa en el temps màxim possible esbiaixat a través dels rellotges en els servidors del sistema. Després de completar les seves escriptures, les transaccions esperen fins que aquesta finestra d'incertesa hagi passat abans de permetre que qualsevol client vegi les dades que s'han escrit.

És desitjable que la finestra d'incertesa sigui el més petita possible, ja que a mesura que es fa més gran, la latència de les transaccions augmenta i la concurrència general del sistema disminueix. D'altra banda necessita assegurar-se que el biaix del rellotge mai sigui superior a la finestra d'incertesa, per tant, les finestres més grans són més segures. Spanner usa una solució *hardware* especialitzada que utilitza GPS i rellotges atòmics.

Pregunta 2 (2 punts)

Enunciat

Donat un SGBD relacional sense cap mecanisme de control de concurrència, suposem que es produeix l'horari següent:

Acció	T1	T2	T3
1	BoT		
2		BoT	
3			BoT
4	R(A)		
5		R(B)	
6			R(C)
7	R(A)		
8	W(A)		
9			R(A)
10		R(B)	
11		W(B)	
12			R(C)
13			W(C)
14	R(C)		
15	W(C)		
16	COMMIT		
17		COMMIT	
18			COMMIT

Assumint que disposem d'un SGBD amb control de concurrència basat en *timestamping* bàsic, on per cada transacció T_i tenim que el *timestamp* de la transacció (TS) és TS(T_i)=i, es demana:

a) Com seria el nou horari amb el control de concurrència descrit? Proporcioneu per a cada acció el RTS (*Read Timestamp*) i WTS (*Write Timestamp*) de cadascun dels grànuls, assumint que tots ells són inicialment buits (valor igual a 0).

b) S'avorta alguna transacció a l'aplicar el control de concurrència descrit? En cas afirmatiu explica breument quines s'han avortat i justifica si és necessari o no l'avortament en base a les possibles interferències.

Nota: la resposta a aquest exercici no ha de superar l'extensió d'una pàgina i mitja.

Criteris d'avaluació

Els criteris d'avaluació que s'aplicaran en la correcció d'aquesta pregunta són els següents:

- Les preguntes no contestades no penalitzen
- Es valorarà la qualitat de la resposta
- Es valorarà la concreció en l'argumentació de la resposta
- Totes les preguntes tenen el mateix pes.
- Es valorarà no superar l'extensió màxima recomanada.

Solució

a) Per poder calcular els horaris d'aquesta taula, cal fer ús del que s'explica en el mòdul 5, a l'apartat de control de concurrència amb *timestamping*.

S'ha completat la taula amb la informació corresponent al *timestamp* de les diferents operacions.

Act	T1	T2	T3	RTS(A)	WTS(A)	RTS(B)	WTS(B)	RTS(C)	WTS(C)
1	BoT								
2		BoT							
3			BoT						
4	R(A)			**1**					
5		R(B)		1		**2**			
6			R(C)	1		2		**3**	
7	R(A)			1		2		3	
8	W(A)			1	**1**	2		3	
9			R(A)	3	1	2		3	
10		R(B)		3	1	2		3	
11		W(B)		3	1	2	**2**	3	
12			R(C)	3	1	2	2	3	
13			W(C)	3	1	2	2	3	**3**
14	R(C) **Abort**			3	0	2	2	3	3
15			**Abort**	0	0	2	2	0	0
16		**Commit**							

b) A l'aplicar el control de concurrència de *timestamp* bàsic, l'acció número 15 provoca un avortament de la transacció T1 donat que el gràNul C que intenta llegir ha estat escrit anteriorment per la transacció T3, que té un TS més gran (TS(T3)=3 → WTS(C)=3 → no es satisfà WTS(C) ≤ TS(T1) necessari per a realitzar l'operació de lectura). L'avortament de la transacció T1 fa que la transacció T3 també avorti, donat que la transacció T1 ha escrit sobre el gràNul A i la transacció T3 el llegeix posteriorment. L'avortament de T1 i T3 fa que calgui restablir els valors anteriors de RTS i WTS de certs gràNuls. De forma més concreta, s'han restablert els valors de RTS i WTS del gràNul A en l'avortament de la T1, i el del gràNul C en l'avortament de la T3.

L'avortament de la transacció T1 evita una interferència de lectura no confirmada (UNCOMMITTED READ) del gràNul C, per la qual cosa resulta necessària. La transacció T3 avorta per motius similars per tal d'evitar també una lectura no confirmada sobre el gràNul A.

Pregunta 3 (3,5 punts)

Enunciat

Un SGBD està processant transaccions i, en un moment donat, el contingut del dietari físic té les entrades següents:

1. 'u', T1, A, 6, 7, nul
2. 'u', T3, C, 5, 4, nul
3. 'u', T2, B, 15, 1, nul
4. cp {T1, T2, T3}
5. 'u' T1, B, 1, 5, 1
6. 'u' T3, G, 5, 9, 2
7. 'u' T4, A, 7, 3, nul
8. 'u' T2, C, 4, 1, 3
9. 'u', T4, I, 3, 2, 7
10. cp {T1, T2, T3, T4}
11. 'u' T2, K, -9, 1, 8
12. 'a' T2
13. 'c', T1
14. 'u', T3, A, 3, -4, 6
15. SYSTEM FAILURE

on les entrades d'*update* tenen el format: 'u', transacció, pàgina, imatge abans, imatge després i apuntador a l'anterior registre d'*update* de la transacció, el qual és nul si és el primer enregistrament d'*update* de la transacció.

Es demana que respongueu als següents apartats:

a) Explica breument, quines possibles estratègies de *checkpointing* podem aplicar en un SGBD. (0,5 punts)

b) Assumint que:

- tenim un gestor de buffers que disposa de 4 *slots* de memòria
- tenim una política STEAL/NO FORCE amb una estratègia d'accions acabades (*action consistent checkpoint*)
- utilitzem una tècnica FIFO per gestionar les pàgines que cal fer *flush*

en quines entrades del dietari i de quines pàgines es farà *flush* si els *checkpoints* es realitzen tal com estableix el diari que dóna l'enunciat? (1 punt)

c) Si tal com s'ha dit, ens trobem en un entorn STEAL/NO FORCE, descriu què farà el procés de *restart* en totes les seves fases quan es produeix la fallada de sistema. (2 punts).

Nota: la resposta a aquest exercici no ha de superar l'extensió de dues pàgines.

Criteris d'avaluació

Els criteris d'avaluació que s'aplicaran en la correcció d'aquesta pregunta són els següents:

- Les preguntes no contestades no penalitzen
- Es valorarà la qualitat de la resposta
- Es valorarà la concreció en l'argumentació
- Es valorarà no superar l'extensió màxima de 2 pàgines recomanada

Solució

a) Explica breument, quines possibles estratègies de *checkpointing* podem aplicar en un SGBD.

Tal i com s'explica en el mòdul 6, hi ha dues possibles estratègies de *checkpointing*: transaccions acabades (*transaction consistent checkpoint*) i accions acabades (*action consistent checkpoint*). Quan seguim una estratègia de transaccions acabades, quan hi ha un *checkpoint*, el sistema suspèn l'execució de transaccions de manera temporal fins que s'acabi l'execució de les transaccions actives, es força l'emmagatzemament físic de la informació, es guarda en el *log* l'operació de *checkpoint* i es permet l'entrada de noves transaccions.

Si l'estratègia és la d'accions acabades, es segueixen els mateixos passos que en l'altra estratègia però no s'espera a que acabin totes les transaccions actives, per tant, s'envia la informació per a emmagatzemar sabent que no totes les operacions s'han confirmat. En aquest cas al guardar l'operació al *log*, s'afegeix la llista de totes les transaccions que estaven actives en el moment del *checkpoint*,

b) En quines entrades del dietari i de quines pàgines es farà *flush* si els *checkpoints* es realitzen tal com estableix el diari que dóna l'enunciat?

Donat que ens indiquen que seguim una estratègia d'accions acabades, caldrà fer *flush* dels *slots* de memòria quan creem un *checkpoint*. A més, donat que utilitzem una política STEAL/NO FORCE, caldrà fer *flush* sempre que haguem de carregar una pàgina en memòria, tots els slots estiguin plens i l'*slot* de memòria en el qual carreguem contingui dades modificades (*dirty bit* a 1).

Si mirem el dietari per determinar quan farem flush: veiem que abans del *checkpoint* de l'acció 4 no es farà cap *flush* donat que no necessitem utilitzar més pàgines que els *slots* de memòria disponibles (4 segons l'enunciat). Al fer aquest *checkpoint* fem *flush* dels 4 buffers, de manera que els canvis que s'han realitzat queden guardats a disc i els slots tornen a quedar lliures (*dirty bit* a 0).

El següent *checkpoint* que trobem és el de l'acció número 10 i fins a aquest punt, s'han modificat 5 pàgines diferents (B, G, A, C i I). Donat que només tenim 4 *slots* de memòria, la modificació de la cinquena pàgina (pàgina I entrada 9) suposarà fer prèviament un *flush* de la primera de les 4 pàgines (pàgina B entrada 5, degut a que utilitzem una tècnica FIFO), per poder fer un *fetch* de la pàgina I i modificar-la.

En el *checkpoint* de l'entrada 10, farem *flush* dels 4 buffers ja que tots ells estaran amb el *dirty bit* a 1. Després no caldrà fer cap més *flush*, ja que no es tornen a utilitzar tots els buffers abans de la fallada del sistema.

c) Si tal com s'ha dit, ens trobem en un entorn STEAL/NO FORCE, descriu què farà el procés de *restart* en totes les seves fases quan es produeix la fallada de sistema.

El procés de *restart* es divideix en 5 accions:

1. *Buffer_and_transaction_list_reset*: reseteja el contingut del *buffer* per tal que no hi hagi informació prèvia a la fallada a cap *buffer*. Les llistes de transaccions de *commit* i d'*abort* també s'esborren.

   CL := buit

   AL := buit

2. *Undo_to_last_CP_phase*: es llegeix el dietari des de la fallada fins a l'últim *checkpoint*, analitzant i processant les entrades, i construint la llista de transaccions confirmades (CL) i avortades (AL). Si l'entrada del dietari es correspon a una operació d'*update*, desfà el canvi si pertany a una transacció no confirmada. D'aquesta manera CL passarà a contenir les transaccions confirmades, mentre que l'AL contindrà les transaccions que han avortat juntament amb les que estaven actives en el moment de la fallada i no han arribat a fer *commit* o *abort*.

   CL := { T1}

   AL := { T2, T3 }

   I els *update* que cal desfer són els de les línies 11 i 14, que corresponen a T2 i T3 respectivament.

3. *Undo_complementary_phase*: completa el procés d'*undo* afegint les transaccions que estaven actives a l'anterior *checkpoint* i que no apareixen a les llistes AL i CL, a més de desfer els canvis que faltin de les transaccions contingudes a AL.

   Mirant les accions que s'han executat fins l'anterior checkpoint, veiem que només cal afegir al T4 a la llista de transaccions avortades.

   CL := { T1}

   AL := { T2, T3, T4 }

   A més cal desfer l'*update* de les línies 2, 3, 6, 7, 8 i 9.

4. *Redo_phase*: cal situar-se de nou a l'últim *checkpoint* i executar els canvis realitzats per les transaccions que estan a la llista de transaccions confirmades (CL). Donat que a CL només tenim la T1 no es tornarà a aplicar cap canvi, donat que no hi ha cap acció de la transacció T1 des de l'últim *checkpoint* i tots els seus canvis ja estan a disc.

5. *Enter_CP*: és l'última acció del procés de *restart* i genera una entrada de *checkpoint* en el *log* per a indicar que la base de dades en aquest punt es troba en un estat consistent. La llista de transaccions actives queda buida.

Pregunta 4 (2 punts)

Enunciat

La base de dades NETFLIX_CLASSICS està formada per les relacions següents, on la clau primària de cada relació està subratllada i les claus foranes estan en cursiva:

Directors (iddirector, name, surname, birthdate, numberfilms, biography, numberawards)

Movies (idmovie, title, plot,*idgenre* ,rate,movieYear)

{idgenre: clau forana a Genre}

Genres (idgenre, name, description)

DirectedBy (idrelation, *idmovie*, *iddirector*, numberawards)

{idmovie: clau forana a Movie}

{iddirector: clau forana a Director}

Suposant que aquesta base de dades està distribuïda seguint l'estratègia de fragmentació següent:

M1 = Movies (movieYear > 2000)
M2 = Movies (movieYear <= 2000)
D1 = Director [iddirector,name,surname, birthdate,biography]
D2 = Director [iddirector,numberfilms]
G1 = Genre (name="comedy")
G2 = Genre (name <>"comedy")
B1 = DirectedBy ⋉ M1
B2 = DirectedBy ⋉ M2

Donada la consulta global següent, trobeu la consulta reduïda basada en fragments que executarà internament el sistema gestor de bases de dades distribuït. Expliqueu els passos principals que seguirà la fase de reducció, assumint que l'estratègia de fragmentació és correcta.

```
SELECT d.name, d.surname,m.title, g.description, b.numberawards
FROM Directors d, Movies m, Genre g, DirectedBy b
WHERE g.name = 'comedy' and m.movieyear <=2000  and m.idgenre = g.idgenre
        and b.idmovie = m.idmovie and d.iddirector = b.iddirector
```

Criteris d'avaluació

Els criteris d'avaluació que s'aplicaran en la correcció d'aquesta pregunta són els següents:

- Es valorarà la qualitat de la resposta
- Es valorarà l'argumentació dels passos de la fase de reducció

Solució

En primer lloc, analitzem l'operació *g.name='comedy'*, en la fase de localització de dades obtenim els següents fragments:

$Q_1 := \sigma_{name='comedy'} (G1 \cup G2)$

Aquesta part de la consulta es pot transformar entrant l'operador de selecció en la unió:

$Q_1 := \sigma_{name='comedy'} (G1) \cup \sigma_{name='comedy'} (G2)$

Substituint en la definició de cada fragment obtenim:

$Q_1 := \sigma_{name='comedy'} (\sigma_{name='comedy'} (G)) \cup \sigma_{name='comedy'} (\sigma_{name<>'comedy'} (G))$

On la segona part resulta en un conjunt buit, per la qual cosa podem simplificar a:

$Q_1 := \sigma_{name='comedy'} (G1) = G1$

Seguidament analitzem l'operació *m.movieyear<=2000*, i obtenim els següents fragments:

$Q_2 := \sigma_{movieyeard<=2000} (M1 \cup M2)$

$Q_2 := \sigma_{movieyear<=2000} (M1) \cup \sigma_{movieyear<=2000} (M2)$

$Q_2 := \sigma_{movieyear<=2000} (\sigma_{movieyear<=2000} (M)) \cup \sigma_{movieyear<=2000} (\sigma_{movieyear>2000} (M))$

quedant reduït a:

$Q_2 := \sigma_{movieyear<=2000} (M2) = M2$

En tercer lloc, ens centrem en la *join* entre *Movies* i *DirectedBy*, que desglossada queda:

$Q_3 := Movies \bowtie DirectedBy = (M1 \cup M2) \bowtie (B1 \cup B2)$

Donada la fragmentació horitzontal derivada, tenim que:

$Q_3 = (M1 \bowtie B1) \cup (M2 \bowtie B2)$

Donada la consulta, l'únic cas que pot retornar valors es l'associat a M2, per la qual cosa resulta:

$Q_3 := M2 \bowtie B2$

Seguidament avaluem el *join* entre *Directors* i *DirectedBy*, donat que la fragmentació de la taula *Directors* és vertical, el *join* queda desglossat de la següent manera:

$Q_4 := DirectedBy \bowtie Directors = (B1 \cup B2) \bowtie (D1 \bowtie D2)$

Al redistribuir la unió amb els *joins* tenim que:

$Q_4 := (B1 \bowtie (D1 \bowtie D2)) \cup (B2 \bowtie (D1 \bowtie D2))$

Donat que només intervé B2, podem simplificar aquesta part de la consulta com:

$Q_4 := (B2 \bowtie (D1 \bowtie D2)) = B2 \bowtie D1$ (ja que volem obtenir el valor *name i surname* de D)

Per últim avaluem el *join* entre *Movies* i *Genres*, que queda:

$Q_5 :=$ Movies $\bowtie$ Genres $= (M1 \cup M2) \bowtie (G1 \cup G2)$

que podem representar com:

$Q_5 = (M1 \cup M2) \bowtie (G1) \cup (M1 \cup M2) \bowtie (G2)$

com que pels valors de la consulta, només obtindrem G1, queda reduït a:

$Q_5 = (M1 \cup M2) \bowtie G1 = M2 \bowtie G1$ (donat que sabem que només tenim M2)

Si reunim totes les parts de la consulta de forma conjunta tenim:

$Q := \prod_{\text{d.name, d.surname,m.title, g.description, b.numberawards}} (Q1 \bowtie Q2 \bowtie Q3 \bowtie Q4 \bowtie Q5)$

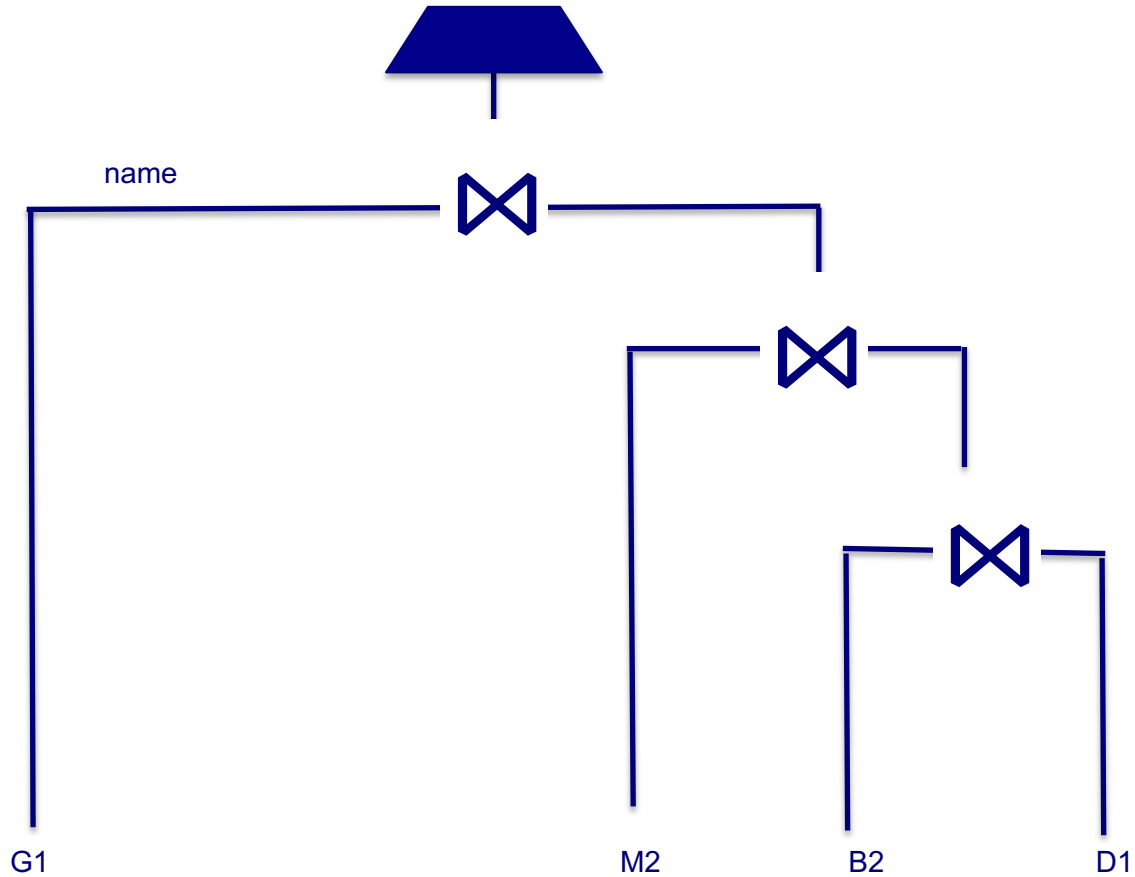
$Q := \prod_{\text{d.name, d.surname,m.title, g.description, b.numberawards}} (G1 \bowtie M2 \bowtie (M2 \bowtie B2) \bowtie (B2 \bowtie D1) \bowtie (M2 \bowtie G1)$

eliminem els elements repetits i ens queda:

$Q := \prod_{\text{d.name, d.surname,m.title, g.description, b.numberawards}} (G1 \bowtie M2 \bowtie B2 \bowtie D1)$

Gràficament obtindríem

Directors.name, Directors.surname,Movies.title, Genre.description, DirectedBy.numberawards



Recursos

Per tal de resoldre aquesta PAC caldrà utilitzar els recursos que s'enumeren a continuació:

- Mòduls 5 (*Transaction models and concurrency control*), 6 (*Data management*) i 7 (*Distributed query optimization*) de l'assignatura Arquitectura de Bases de Dades.
- "NewSQL database systems are failing to guarantee consistency, and I blame Spanner" de Daniel Abadi (http://dbmsmusings.blogspot.com/2018/09/newsql-database-systems-are-failing-to.html)

Criteris de valoració

A l'enunciat de cada exercici s'indica el valor del mateix sobre la puntuació total de la PAC. Aquesta activitat representa el 50% de la nota d'avaluació continuada de l'assignatura i la seva realització és **obligatòria** per tal de superar l'assignatura.

No s'acceptaran ni es tindran en compte els lliuraments realitzats fora dels terminis indicats al calendari de l'aula.

Format i data de lliurament

Cal lliurar, a través de la bústia de lliuraments de l'aula, un arxiu .pdf amb les respostes a l'enunciat i el nom del qual contingui el codi de l'assignatura, el vostre cognom i el vostre nom, així com la indicació de que es tracta de la PAC2. La resolució i lliurament d'aquesta activitat només es pot fer de forma individual.

La data límit per lliurar la PAC2 és el **14/01/2020**.