Replicarea Bazelor de Date

Sisteme de Operare Avansate

Stoica Serghei

Cuprins

1. Întroducere
   1. Replicarea și Backup-ul.
   2. Replicarea și Caching.
   3. Replicarea
2. Mecanizme de replicare si sincronizare
   1. Replicarea imediată (Eager replication)
   2. Replicarea leneșă
3. Tipuri de replicare
   1. Replicarea snapșot
   2. Replicarea prin imbinarea datelor (merge replication)
   3. Replicarea tranzacțională
   4. Replicarea în PostgreSQL
   5. Replicarea în Microsoft SQL Server.
   6. Replicare în bazele de date Cassandra DB.
4. Concluzii
5. **Întroducere. Replicare, Backup, Caching.**

Odată cu dezvoltarea tehnologică, tot mai multe date se pastrează în format electronic. Aplicațiile de calculator existente produc o cantitate din ce in ce mai mare de informație care trebuie pastrată și organizată. Bazele de date au rolul de a pastra informația intro forma mai mult sau mai puțin organizată. Dacă la inceput bazele de date trebuiau să stocheze o cantitate relative mică de informație, astăzi trebuie să stocheze zeci de GB. Există mai multe familii de baze de date, fiecare avînd anumite avandaje specifice, dar în general putem vorbi de baze de date relaționale SQL și baze de date NoSQL. Datorită cantității mari de informație care se dorește a fi stocată, în general, o bază de date pote fi distribuită pe mai multe noduri. O bază de date distribuită este caracterizată și trebuie să asigure: consistența datelor, disponibilitate datelor, și să fie tolerant la partiționare. Conform teoremei CAP aceste 3 caracteristici nu pot fi asigurate concomitent. Prin consistență se ințelege faptul că toate nodurile văd aceleași date. Prin valabilitate se ințelege faptul că datele sunt disponibile, iar baza de date este tolerantă la defectarea unor noduri. Prin toleranță la partiționare se înțelege faptul că baza de date poate fi distribuită mai multe noduri, ideal pe oricîte.

Aplicațiile de bussines au cerințe destul de exigente asupra bazelor de date. Datele trebuie să poată fi restabilite în urma avarii în urma caruia sistemul este indisponibil. Sistemul trebuie să fie tolerant la defectarea unor noduri singulare. Sistemul trebuie să asigure timpi de răspuns cît mai mici posibili. Pentru a asigura cerințele necesare există mai multe tehnici partajare, copiere și sincronizare a datelor între nodurile pe care este distribuită baza de date.

Replicarea. Prin replicare se înțelege procesul prin care se copie și se sincronizează datele între diferite noduri peste care este distribuită baza de date. Principalul scop al replicării este de a asigura consistența datelor dintre noduri.

Backup. Prin backup se întelege înțelege crearea unei copii totale a datelor care poate fi folosită în cazul în care sunt pierdute datele actuale. Spre deosebire de replicare în care copierea datelor are ca scop asigurarea consistenței datelor, procedura de backup are ca scop restabilirea datelor în caz de avarii. Backup-up reprezintă un istoric al datelor.

Caching. Prin caching se înțelege memorarea ultimilor date folosite în ideia ca există o probabilitatea mare ca aceste date să fie refolosite și se poate reduce timpul de acces la aceste date.

* 1. Replicarea și Backup-ul.

Mai sus au fost enumerate mai multe tehnologii care fac copii a datelor pentru a crește o caracteristică a sistemului. Pentru a crește performanța și fiabilitatea sistemului trebui aplicate toate metodele de mai sus. Replicarea are ca scop asigurarea sincronizării datelor între noduri, care cu cît sunt mai multe cu atît asigură o disponibilitate mai bună a datelor. Backup-ul trebui doar să asigure posibilitatea de recuperare a datelor în urma unui dezastru. Prin disponibilitate se în perioda de timp în care aplicația (baza de date) funcționează în condiții normale raportată la perioada totala de timp în care funcționează aplicația. Baza de date este distribuită pe mai multe noduri. Din anumite cauze un nod poate deveni indisponibil la un anumit timp. Dacă baza de date este doar pe un singur nod, dacă singurul nod devine indisponibil atunci tot sistemul devine indisponibil. Dacă baza de date este distribuită, atunci sistemul continuie să fie disponibil. Crește performanța sistemului, prin performanță înțelegîndu-se numărul de cereri care pot fi procesate. Cererile către sistem se procesează de către toate nodurile disponibile. În acest context apare întrebarea consistenței datelor, adică datele procesate de catre un nod trebuie să fie cunoscute și de către celelalte noduri. Orice nod din sistem poate dispare întrun moment dat la fel cum un nod nou poate să apară, conform principiului consistenței, nodul nou trebuie să ofere aceleși date ca și celelate. În acest context tehnologia de replicare asigură sincronizarea datelor dintre noduri, care la rîndul lor asigură disponibilitatea datelor și performanța sistemului.

Prin backup se înțelege copierea datelor si salvarea lor cu scopul dea putea fi restabilite în cazul în care se pierd datele actuale. Datele backup se referă la un moment concret de timp și pot fi considerate istoric, fiind o copie secundară de date. Procesul de copiere a datelor poate fi repetat periodic automat, sau poate fi facut la comandă. Backup-ul este o componentă cheie a unei strategii bune de protecție a datelor. Din datele backup se poate restabili total informația, însă aceasta este in starea la care sa facut copierea datelor. Avînd în vedere ca se face o copie completă a datelor, procesul de buckup nuu poate fi rulat la fiecare schimbare de date. Se definesc noțiunile de RPO (recovery point objective) și noțiunea de RTO (recovery time objective). Prin RPO se înțelege cu cît sunt învechite datele backup. Uneori poate fi necesar un RPO de 0 iar alteori, în alte cazuri, în funcție de aplicație poate fi acceptat un RPO de ore sau chiar zile. Prin RTO se înțelege timpul total necesar pentru restabilirea datelor din fișierele backup. Este de dorit ca RTO sa fie cît mai mic, adică datele să fie restabilite cît mai repede, ideal 0.

Replicare vs Backup

|  |  |  |
| --- | --- | --- |
|  | Replicare | Backup |
| Definiție | Presupune sincronizarea datelor prin copierea lor între diferite noduri. Tote datele sunt considerate ca fiind aceleași, orice schimbare întro replică propagînduse și în celelalte. | Presupune copierea datelor, este legată de un moment de timp. |
| Cerințe la nivel de resurse | Tehnologia de replicare necesită cîte o structură similară pentru fiecare replică, adică cîte un nod. | Backup-ul necesită doar spațiu suplimentar pentru stocarea copiei datelor, copia datelor nefiind operațională. |
| Scopul | Tehnologia de replicare are ca scop creșterea fiabilității sistemului, disponibilității datelor și perfomanței sistemului. | Backup-ul are ca scop readucerea sistemului la o stare anterioară în cazul în care se întîmplă avarii în urma carora datele sunt pierdute. |
| Implicații economice | Este orientat pe aplicații de bussines și trebui să asigure continuitatea funcționării aplicației în cazul în care „cad” noduri individuale. Este mult mai scup decît backup-ul. | Este o modalitate relativ ieftină de a evita pierderile de date. Nu asigură continuitatea funcționării sistemului. |

* 1. Replicarea și Caching.

Caching este o tehnologie prin care datele maifregvent utilizate sunt copiate întrun depozit mai rapid co scopul de mari viteza de raspuns a sistemului. Spre deosebire de replicare sau backup unde avem cel puțin o copie copletă a tuturor datelor, caching-ul are o copie a doar o parte din date, cele mai fregvent utilizate. De obicei cach-ul se stochează în RAM, ceia ce permite o viteză de acces mare. La fel ca bazele de date, cachul poate fi distribuit peste mai multe noduri. Dimensiunea este variabilă în funcție de memoria disponibilă. Este un layer peste alte depozite de date mai lente. După cum sa menționat, în cache sunt stocate doar o parte din date. Daca anumite date nu sunt găsite în cache atunci sunt cautate mai întai în cachurile nodurilor vecine, iar dacă nu sunt gasite nici acolo atunci datele sunt cautate în depozitul original de date. Cachul suportă operații simple de inserare și extragere a datelor. Este necesar ca la schimbarea unor date chachul sa fie golit pentru a evita incosistența datelor. Spre deosebire de replicare care are rolul de a crește atît disponibilitatea datelor cît și performanța aplicației, cachul crește doar performanța aplicației.

* 1. Replicarea

Replicarea este un proces care constă în realizarea şi distribuirea de copii ale datelor şi, în plus, permite ca modificările efectuate să fie propagate în mod consistent la copiile corespunzătoare. Distribuirea acestor replici are ca scop procesarea datelor la nivel local.

Procesul de replicare sporeşte securitatea sistemului şi îmbunătăţeşte viteza operaţiunilor de procesare de date. În afară de aceasta, aplicaţia poate funcţiona chiar şi dacă un server local ar eşua, dar alte servere cu baze de date replicate rămîn accesibile. În cazul replicării datelor, este mai dificil să se asigure menţinerea consistenţei acestora pentru că actualizarea unui fragment dintr-o bază de date trebuie să fie propagată la toate copiile fragmentului din bazele de date replicate. Decizia replicării unui fragment este influenţată de raportul dintre numărul de procese de scriere şi citire ale fragmentului. Copierea datelor pe mai multe servere concomitent oferă un paralelism de procesare. Marea problema pe care apare la replicare este timpul de propagare a noilor date catre celelalte noduri, timp care trebuie să fie cît mai mic, aceasta impune cerințe mari asupra sistemului hardware și software. Pentru a asigura replicarea, în sistemele în timp real, fiecare nod rulează o buclă de sincronizare care poate comunica cu nodurile vecine dar și cu alte servere. Teortic, în urma replicării de k ori, sitemul poate fi văzut din exterior ca un sistem care are o singură copie a datelor cu un hard care este mai puternic de k ori. În acest context putem spune că am obținut o scalabilitate verticală. Prin scalabilitate verticală se înțelege creșterea performanței sistemului prin creșterea performanței sistemului hard. Viceversa, este scalabilitatea orizontală, în care creșterea performanței sistemului este făcută pe baza creșterii numărului de mașini pe care rulează sistemul. În ambele cazuri, scalabilitate orizontală/scalabilitate verticală, performanța nu crește liniar cu creșterea performanței hardului/numărului de mașini.

Replicare poate să difre în fucție de gradul de replicare a datelor. O bază de date poate fi:

* *Nereplicată* – sistemul are o bază de date care se află întrun singur nod
* *Parțial replicată –* este atunci cînd există date partajate care sunt replicate, și date specifice fiecărui nod
* *Totatl replicată –* toată datele sunt replicate pe mai multe noduri.

În sistemele ce se doresc a fi performante se optează pentru replicarea totală a datelor, dar a în funcție de distribuția geografică a nodurilor și specificul aplicație, la fel de bine se poate opta pentru o replicare parțilă.

În ceia ce privește replicare în sistemele real-time, trebuie ținut cont de cîteva aspecte importante:

* *Consistența –* Fiecare server își poate sincroniza replicile sale întrun anumit mod, ceia ce determină un anumit nivel de consistență. Trebuie de facut un compromis între nivelul de consistență dorit și disponibilitatea datelor. Un nivel de consistență mai mare înseamnă un overhead de sincronizare mai mare, ceia ce implică o performanță mai mică.
* *Responsivitatea –* prin responsivitate înțelegem timpul de reacție la o cerea a utilizatorului. În general timpul de raspuns a aplicațiilor ce au baze de date trebuie să fie între 50-100 ms pentru a avea o performanță optimă. Prin replicare se pot aduce datele mai aprope de utilizator, micșorînd timpul de răspuns al aplicației.
* *Scalabilitatea –* Prin scalabilitate se înțelege posibilitatea de a extinde aplicația pe mai multe mașini cu scopul de a acomoda mai mulți utilizatori. Cum sa explicat anterior replicarea oferă o scalabilitate pseudo verticală.

1. **Mecanizme de replicare si sincronizare**

Replicarea datelor este un concept fundamental în sistemele de baze de date distribuite care a fost studiat o periodă lungă de timp. Au fost dezvoltate mai multe tehnici de replicare pentru a crește scalabilitatea, fiabilitatea și performanța sistemelor distribuite. Aceste mecanisme trebuie să țină cont de dimensiunea datelor cu care lucrează sistemul, de numarul de noduri și performanța fiecărui nod, de lațimea de bandă pe care o au nodurile pentru a comunica între ele. Au fost dezvoltate tehnici de replicare activă și pasivă cu modalitate de sincronizare imediată și respectiv „leneșă”. Fiecare tehnică de repicare are caracteristicile sale specifice. Conform teoremei CAP, nu putem avea consistență, disponibilitate și toleranță la partiționare concomitent. Trebuie să facem un compromis între aceste trei caracteristici în funcție de aplicație. Întrun sistem bancar am dori să avem consistență maxim și să facem optimizarea între celelalte 2 caracteristici. Întrun sistem de distribuție de fișiere mai degrabă am vrea să avem toleranță la partiționare, ceia ce ar permite distribuirea datelor pe mai multe noduri.

Există 2 criterii de clasificare a mecanismelor de replicare:

* Modul în care se sincronizează datele și selectarea datelor care trebuie sincronizate
* Modul de selecție a nodurilor care trebuie să primească datele ce trebuie sincronizate
  1. Replicarea imediată (Eager replication)

Prin replicare rapidă se înțelege faptul că odată ce un nod master a suferit niște schimbări de date, nodul master are grijă să inițializeze imediat procesul de sincronizare a datelor. Tote nodurile replici primesc date pentru a improspăta baza de date pe care o dețin. Acest tip de sincronizare folosește tranzacții imbricate bazate pe protocole pentru tranzacții distribuite care asigură atomicitatea și consistența datelor. Overheadul de sincronizare este mult mai mare ca în cazul replicării leneșe.

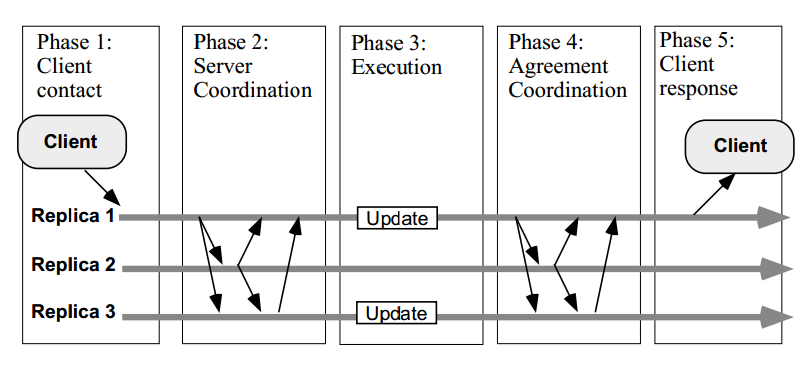


Figura 1. Schema de sicronizare rapidă [ VI ]

Din schema de mai sus observăm că clientul nu primește confirmare pînă cînd datele nu sunt copiate pe tote replicile necesare. Clientul fie execută tranzacția pe toate replicile, fie pe niciuna. Aceasta introduce un timp de raspuns foarte mare.

* 1. Replicarea leneșă

Nu se face copierea imediată a datelor pe toate replicile. Clientul primește un raspuns de confirmare de la nodul master, iar nodul master iși ia responsabilitatea să propage schimbările și pe replicile necesare dar peste un anumit timp. Acest mecanism de replicare scade timpul de răspuns al aplicației, dar necesită algoritmi mai coplecși de sincronizare care să țină cont de eventuale conflicte ce pot apărea între date care au fost modificate pe noduri diferite. În acest sens se poate folosi algoritmul Lamport. Avantajul replicării leneșe este creșterea responsivității sistemului cu prețul reducerii consistenței.

1. **Tipuri de replicare**

Pot fi următoarele tipuri de replicare:

* *Replicarea snapshot –* Se copie datele integral de pe un server pe altul, sau dintro baza de date în alta.
* *Replicarea prin îmbinarea datelor –* datele din mai multe baze de date sunt imbinate și compiate întro bază de date.
* *Replicarea tranzacțională –* este creată o replică inițială de tip snapshot după care se primesc update-uri periodice în funcție de schimbarile survenite.
  1. Replicarea snapshot

Este una dintre cele mai simple metode de replicare. Replicarea periodică funcționează prin unor blocuri mari de date. Poate fi folosită atunci cînd replica funcționează întrun mod readOnly sau cînd se poate funcționa fară a avea cele mai recente date. Timpul dintre doua update-uri consecutive se numește latență, și în această perioadă se lucrează cu date care pot fi false.

Acest tip de replicare se poate folosi pentru date care se schimbă rar și pe lîngă asta ne permitem să existe o probabilitate ca aceste date să fie eronate. De exemplu poate exista o bază centrală a unei companii cu inventarul companiei, și săptămînal să fie facută replicarea pe bazele de date din filialele companiei. Aceste date sunt relativ constante și în cazul unor erori nu provoacă mari pierdieri.

Acet tip de replicare poate fi folosit în soluțiile în care conexiunea dintre noduri este instabila. Replicarea snapshot, este singurul tip de replicare care poate fi folosit pentru date care sunt organizate în tabele fără cheie primară.

La replicarea snapshot nodul master crează niște fișiere numite snapshot-uri, care le distribuie către celelalte noduri. Din fișierele snapshot sunt extrase datele și introduse în bazele de date respective.

Cazuri în care poate fi folosită replicarea snapshot:

* Datele sunt relativ statice
* Este acceptabil să avem date nesincronizate
* Cînd volumul de date care se modifică e mare dar se întîmplă rar
  1. Replicarea prin imbinarea datelor (merge replication)

Replicarea prin imbinarea datelor permite ca nodul care publică datele să salveze pentru fiecare schimbare intervenite un snapshot în cazul în care nu există legătura între cu nodurile replici, iar poi cînd legătura este restabilită se extrag datele din snapshoturi care se combină și se transmit către nodurile replici.

Acest tip de replicare permite nodurilor să lucreze off-line ca mai apoi să se poată face sincronizarile necesare.

Avînd în vedere că se pot acumula un numar mare de date nesincronizate, pot apărea conflicte între versiuni diferite de date de pe noduri diferite. În acest sens este necesară o modalitate de arbitrare a conflictelor. Modalitățile de rezolvare a conflictelor sunt configurabile. Acest tip de replicare este cel mai complex tip de replicare deoarece permite atît nodului sursă de date cît și nodului replică destinație să facă schimbări pe aceleași date. Nodurile pot lucra independent o anumită perioadă de timp.

Replicarea prin îmbinarea datelor este utilă cînd:

* Aceleși date se schimbă pe noduri diferite și necesită să fie aleasă o singură versiune.
* Există perioade în care nodurile sunt off-line
* Apar un numar acceptabil de conflicte și în cazul în care apar ne permitem să fie violate conceptele ACID.
  1. Replicarea tranzacțională

Poate fi considerată opusul replicarii de tip snapshot. Sincronizarea este facută imdiat ce apar schimbări de date. Fiecare tranzacție care apare este imediat replicată și pe nodurile necesare. Poate fi cofigurată și opțiunea de a acumula datele de la mai multe tranzacții și să fie transmise toate odată. Acest tip de replicare necesită o conexiune bună între noduri, bandă destul de largă și principalul latență mică. Tranzacțiile se stochează întrun fișier de Log-uri. Dacă conexiunea pică acest fișier poate crește rapid.

Replicarea tranzacțională începe cu un snapsot al datelor inițiale. Datele din snapshot sunt alterate apoi prin aplicarea tranzacțieilor. Se poate seta perioda de sincronizare a snapshotului.

Odată ce a fost creat snapshotul datelor inițiale, replicarea tranzacțională se face cu ajutorul unui agent care citește fișierele Log, extrage tranzacțiile și le stochează întro bază de date distribuită. Un alt agent de distribuire extrage tranzacțiile din baza de date distribuită și le transmite catre nodurile destinație unde fiecare nod își updatează snapshotul inițial.

Replicarea tranzacțională este utilă cînd:

* Infrastructura ne permite, adică avem o conectare stabilă a nodurilor
* Dorim ca schimbarile să se propage catre nodurile destinație imediat ce apar, adică să asigurăm o consistență sporită
* Toate copierile de date au caracteristici ACID

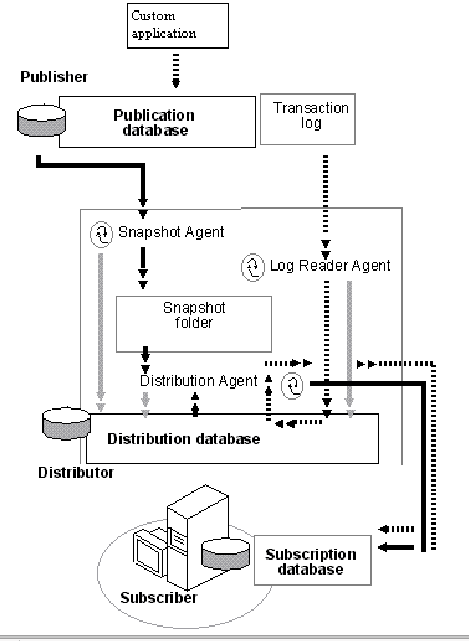


Figura 2. Schema replicației tranzacționale [V]

* 1. Replicarea în PostgreSQL

Există mai multe tool-uri pentru replicarea bazelor de date PostgreSQL printre care: Usogres, eRServer, PostgreSQL Replicator, Postgres-R. Postgres-R este o extensie PostgreSQL care oferă metode eficiente de replicare a bazelor de date cluster, totodată asigurînd consistența. Este folosită în primul rînd pentru echilibrarea/distribuirea încărcării sistemului și disponibilitate sporită a sistemului. Postgres-R asigură o scalabilitate a procesului de replicare, este mai fiabil și flexibil în comparație cu sistemele centralizate de replicare.

Postgres-R poate replica obiecte mari și folosește replicare asincronă de tipul store and forward. Se pot adăuga sau scote noduri în timpul rulării sistemului fără a fi nevoie de a fi oprit. Nodurile picate sunt detectate și scoase din sistem fară ca sistemul să fie afectat. Postgres-R folosește algoritmul 2PL pentru controlul tranzacțiilor concurente. Tranzacțiile sunt de tipul read-one-write-all și sunt execută în 4 faze. În prima fază tranzacția se execută local, pe nodul pe care a venit cererea. În faza a doua se propagă schimbările de date. Faza a treia este faza de sincronizare, în care nodurile replici primesc tranzacția. În faza a patra tranzacția este executată pe nodurile replici.

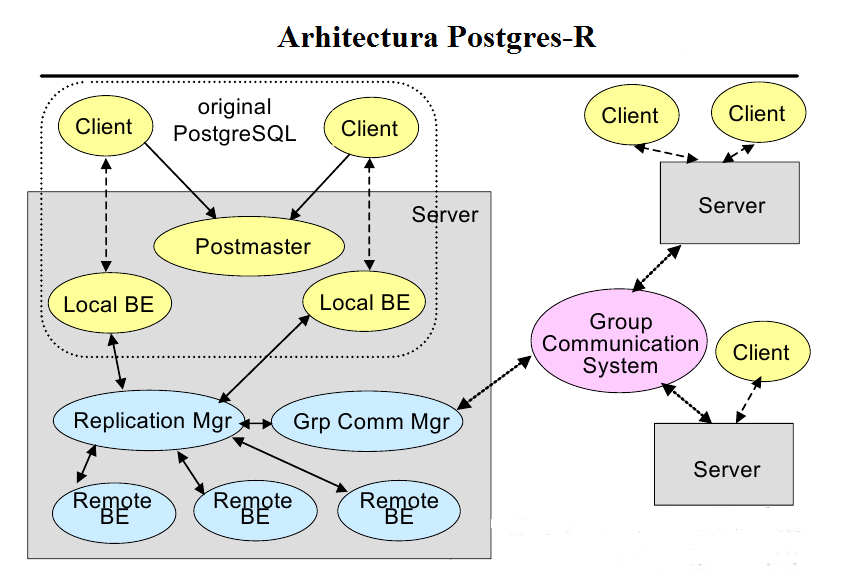
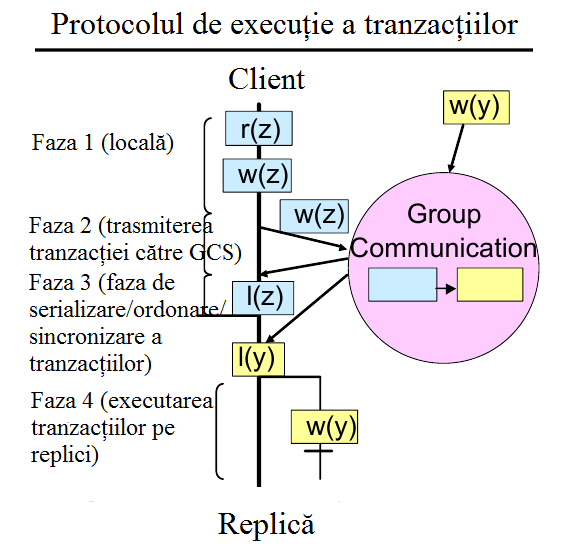


Figura 3. Arhitectura Postgres-R [III]

Pe fiecare server rulează cîte o instanță de Postgres-R. Cînd un client vrea să acceseze baza de date, trimite cererea către Postmaster. Postmasteru crează cîte un proces pentru fiecare bază de date locală pe care vrea să o acceseze clientul, după care se stabilește conexiunea dintre baza locală și client. Clientul se poate conecta la orce server disponibil. Tranzacțiile se execută în patru pași. Manageru de replicare „Replication Mgr” este responsabil de procesarea mesajelor externe.



Replicarea tranzacțională în PostgresR [ VII ]

* În primă fază tranzacțiile sunt executate local.
* În faza a doua sunt transmise operațiile care necesită scrieri de date către Group comunication system.
* GCS serializează mesajele primite stabilind între ele o ordine totală rezolvînd conflictele, după care le transmite către nodurile replici.
* În faza a 4 mesajele sunt execute pe fiecare replică în parte.

Avantaje Postgres-R

* Performanță ăn sesnsul de timpi de răspuns și rata datelor.
* Asigură consistență sporite și fiabilitatea sistemului (disponibilitate)
* Este construit peste SGBD PostgreSQL, deci este un modul în locul caruia poate fi folosit oricare altul, și nu depinde de versiunea PostgreSQL.
  1. Replicarea în Microsoft SQL Server.

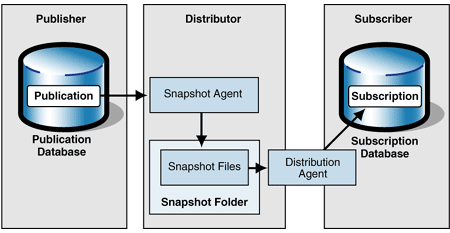
Tehnologia Microsoft SQL Server oferă o suită de unelte specifice lucrului cu bazele de date. Acestea vin toate la pachet cu sistemul clasic de gestiune a bazelor de date. SGBD-ul microsoft pune la dispoziție o varietate de tehnologii de replicare a bazelor de date. Aceste tehnologii sunt încorporate în SGBD-ul standart. SGBD-ul microsoft SQL server implimentează toate cele 3 tehnologii de replicare descrise în capitolul 3: replicarea snpșot, replicare prin combinarea datelor, replicarea tranzacțională.

Replicarea tranzacțională este folosită de obicei în scenarii peer-to-peer avînd nevoie de o conectare între noduri cu lățime de bandă mare. Implimentarea celor 3 tipuri de replicare se regăsește în SQLCE 3.5 și SQLCE 4.0 care este compatibilă de la Microsoft Windows Server 2012 în sus, inclusiv și cu sistemul de operare Windows 8.

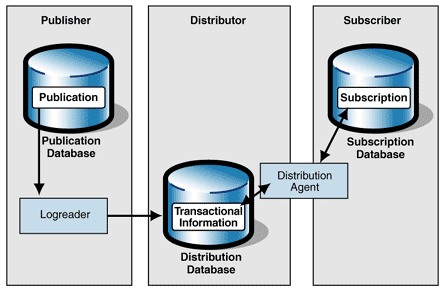
O alternativă pentru sistemul de replicare încorporat este utilitarul Microsoft Sync Framework care are rolul de sincronizare a datelor.

Sistemul de replicare de la microsoft poate interacționa și cu alte tipuri de SGBD-uri, precum cel de la Oracle.

Tehnologia microsoft este foarte flexibilă și permiteimplimentarea unei varietați mari de tipuri de replicare pecînd de la cele trei tehnologii de bază.



Replicare snapșot pentru microsoft SQL server. [V]



Replicarea Tranzacțională pentru microsoft SQL server. [V]

După cum vedem, toate tipurile de replicare urmează acelașii patern în care avem: Nodul sursă de date, un distribuitor a datelor și nodurile destinație pentru date adică replici. În secțiunea de distribuitori avem o varietate mare de implimentări. La fel în secțiunea de subscriber pot fi și sisteme eterogene precum SGBD-ul Oracle sau IBM DB2. Este un sistem de replicare foarte flexibil putînduse alege orice metodă de replicare precum și ce replici și unde replici ajungînduse la o rezoluție de obiect a bazei de date.

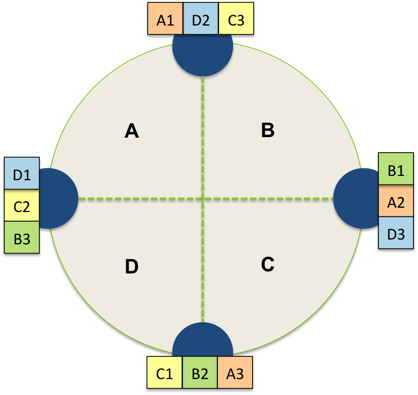
* 1. Replicare în bazele de date Cassandra DB.

Baza de date Cassandra DB este un sistem de tip NoSQL column family. Este proiectat pentru stocarea unor dimensiuni mari de date. Avînd în vedere acest lucru, baza de date este distribuită și deci trebuie asigurată o disponibilitate a datelor și toleranța la partiționare.

În sistemul cassandra, fiecare rînd al unui tabel are replici. Replicile sunt identificate pe baza indentificatorului de rînd. Numărul de replici este un parametru configurabil, precum și algoritmul de distribuție a replicilor în cluster. Numar total de replici este un parametru configurabil și se numește factor de replicare. Un factor de replicare de 1 ar însemna defapt că datele nu sunt replicate. Factorul de replicare nu poate fi mai mare decît numărul total de noduri în cluster.

Există doua strategii de distribuire a replicilor:

* *Strategia simplă –* Este folosită un singur cluster întrun singur centru de date. Este strategia implicită pentru SGBD Cassandra la crearea unui keyspace. Conform strategiei simple, datele sunt scrise pe nodul determinat de partiționer. Apoi fiecare replică este scrisă pe următorul nod în sensul acelor de ceasornic ca în imaginea de mai jos, in care datele A1,B1,C1 și respectiv replicile lor A2,B2,C2 și A3,B3,C3.



Strategia de replicare simplă în CassandraDB [ IV ]

* *Strategia de replicare Network Topology* – este folosită cînd avem un cluster întins peste mai multe centre de date. Strategia specifică cîte replici avem în fiecare centru de date. Pentru a decide cîte replici trebuie să avem în fiecare centru de date, trebuie să ținem cont de doi factori. Trebuie să satisfacem cererile de citire fără a crește latența datorată comunicării dintre centre de date, și trebuie de ținut cont de scenariul în care nodurile pică. Se pot opta petru 2 replici per data centru. În acest caz dacă una din replici devine indisponibilă, se mai pot citi datele din datacentru local cu un nivel de consistență 1. Se pot opta și petru 3 replici per data centru ceia ce ar asigura o toleranță la defecte mai mare. Strategia de replicare NetworkTopology determină plasarea replicilor în fiecare centru de date. Conform acestei strategii, datele inițiale sunt plasate în nodul ales de partiționer. Apoi urmatoarea replică este plasată în nodul următor de rang diferit, parcurgînduse în sensul acelor de ceasornic, sau dacă nu e gasit se plasează în noduri de acelașii rang. Nodurile sunt grupate după caracteristici tehnice și fizice similare, deci 2 noduri cu acelașii rang au aproximativ aceiași probabilitate de a pica. Deci strategia are grija ca să nu plaseze replici în noduri care pot fi plasate pe același suport fizic și pot pica concomitent.

1. **Concluzii**

După cum sa spus mai sus, replicarea bazelor de date nu e un proces simplu, dar odată aplicat poate crește considerabil performanțele sistemului. Crește valabilitatea datelor, viteza sistemului, scalabilitatea și toleranța la defecte.

Tote acestea desigur au un preț care se traduce în primul rînd prin mai mult echipament hardwareprecum spațiu de stocare sau chiar un nod întreg care este echivalent cu o mașină (fizică sau virtuală). Pe lîngă asta, conform teoremei CAP, pierdem din cosistența datelor.

Sunt mai multe tehnici de replicare. Tehnica de replicare trebuie aleasă cu grijă în funcție de tipul aplicației, performanțele dorite, și nu în ultimul rînd, resursele disponibile.

**Referințe :**

1. Analele Universităţii “Constantin Brîncuşi” din Tîrgu Jiu, Seria Economie, Nr. 4/2010 - REPLICAREA DATELOR ОN MEDIILE DISTRIBUITE
2. International Journal of Recent Trends in Electrical & Electronics Sept 2011. ISSN: 2231-6612. Replication: Analysis & Tackle in Distributed Real Time Database System SANJAY KUMAR TIWARI, A.K.SHARMA, V.SWAROOP Department of Computer Sc. & Engg. M.M.M. Engineering College, Gorakhpur, U.P., India.
3. International Journal of Computer Applications (0975 – 8887)Volume 13– No.6, January 2011. Database Replication: A Survey of Open Source and Commercial Tools.
4. <http://www.datastax.com/docs/1.0/cluster_architecture/replication> accessat la data: 24/01/2015
5. Biblioteca microsoft MSDN https://msdn.microsoft.com accesat la data 25/01/2015
6. Understanding Replication in Databases and Distributed Systems. M. Wiesmann, F. Pedone, A. Schipe, B. Kemme, G. Alonso.
7. PostgreSQL Database Replication Options. Darren Johnson