REPUBLIKA E SHQIPËRISË
UNIVERSITETI I TIRANËS
FAKULTETI I SHKENCAVE TË NATYRËS
DEPARTAMENTI I INFORMATIKËS

JULIAN FEJZAJ

"ARKITEKTURAT E BAZAVE TË DHËNAVE TË ORIENTUARA NGA SHËRBIMET"

TEZË DOKTORATURE

UDHËHEQËS SHKENCOR: PROF.ASOC. ENDRI XHINA

TIRANË, 2014

UNIVERSITETI I TIRANËS
FAKULTETI I SHKENCAVE TË NATYRËS
DEPARTAMENTI I INFORMATIKËS

DISERTACION

i paraqitur nga:
MSc. Julian Fejzaj

udhëhequr nga
Prof. Asoc. Endri Xhina

Për marrjen e gradës shkencore

DOKTOR

Specialiteti: Informatikë

Tema:

"Arkitekturat e Bazave të Dhënave të Orientuara nga Shërbimet"

Mbrohet më dt. …./...../201.. para jurisë:

1. .....…..…….. **Kryetar**
2. ……..... **Anëtar (Oponent)**
3. ……..... **Anëtar (Oponent)**
4. …… **Anëtar**
5. …… **Anëtar**

Kapitulli 1 - Prezantim me Arkitekturën e Orientuar drejt Shërbimeve (SOA).... 1
1.1 Arkitektura e Orientuar drejt Shërbimeve (SOA)..... 1
1.1.1 Përbërja, Ndërtimi, Infrastruktura..... 1
1.1.2 Zhvillimi i SOA..... 2
1.1.3 Karakteristikat kryesore të SOA..... 3
1.2 Komponentet arkitekturore të SOA 6
1.3 Avantazhet, Sfidat..... 7
1.3.1 Avantazhet e SOA..... 7
1.3.2 Sfidat e SOA. 7
Kapitulli 2 . Integrimi i Aplikimeve 8
2.1 Çfarë është Integrimi i Aplikimeve?..... 8
2.2 Nevoja dhe kërkesa e tregut për Integrimin e Aplikimeve..... 9
2.3 Mënyrat e Integrimit të Aplikimeve 10
2.3.1. 'Platform Integration'..... 11
2.3.3 'Data Integration' 11
2.3.4 'Component Integration' 12
2.3.2 'Application Integration' 12
Kapitulli 3 Arkitektura e Integrimit të Aplikimeve..... 16
3.1 Arkitektura 'Point-to-Point' 16
3.2 Arkitektura 'Middleware'..... 17
3.3 Stilet e integrimit..... 18
3.4.1 'File Transfer' 19
3.4.2 'Shared Database'..... 19
3.4.3 'Remote Procedure Call' 20
3.4.4 'Messaging'..... 21
3.4.5 'Web Services' 21
Kapitulli 4 – Ndikimi i rrjetit në Bazat e të Dhënave të Shpërndara 23
4.1 Rëndësia e rolit të rrjetave në arkitekturën SOA..... 23
4.2 Bazat e të dhënave të shpërndara 24
4.2.1 Bazat e të dhënave Homogjene dhe Heterogjene 24
4.3 Vonesat në rrjet..... 25
4.3.1 Vlerësimi i Algoritmit të Rrugëzimit..... 26
Kapitulli 5 – SOA, Virtualizimi dhe Cloud 27
5.1 Virtualizimi 27
5.1.1 Virtualizimi i shërbimeve..... 28
5.1.2 Historiku e virtualizimit të shërbimeve..... 28

5.2 Cloud computing 30
5.2.1 Arkitektura Cloud Computing 30
5.2.1 SOA dhe Cloud Computing 33
5.2.2 Lidhja midis SOA dhe Cloud Computing 34
5.3 Cloud Computing dhe Virtualizimi 35
5.4 Teknika e Migrimit Live 35
5.4.1 Migrimi live Hyper-V 36
5.4.1 Optimizimi i Migrimi live 36
Kapitulli 6-Arkitekturat e Bazave të Dhënave të Orientuara nga Shërbimet (SODA) 39
6.1 Prezantim me Arkitekturën SODA 39
6.1.1 Përdorimi i SODA 41
6.1.2 Të dhënat në SODA 43
6.2 Mjete të Ms SQL Server për implementimin e koncepteve SODA 46
6.2.1 Suporti i XML 47
6.2.2 Native Web Services 47
6.2.3 Gjuha CLR dhe integrimi me SQL Server 50
6.2.4 Database Change Notifications – njoftimet e ndryshimeve në bazën e të dhënave 52
Kapitulli 7 – Arkitektura e Shërbimeve të Mesazheve dhe Radhëve 54
7.1 Shërbimet Broker të Mesazheve të bazuara në radhë në SQL Server 54
7.2 Arkitektura Service Broker 55
7.3 Ndërtimi i aplikacioneve i bazuar në radhën e mesazheve duke përdorur shërbimet Broker në Sql Server. 60
Kapitulli 8– Shkëmbimi i të dhënave në mënyrë standarte 76
8.1 Modeli Relacional dhe XML 76
8.1.1 Roli i XML 76
8.1.2 XML dhe të Dhënat Relacionale 76
8.1.3 XML në Sql Server 79
8.2 Kalimi nga te Dhënat Tabelare në XML 81
8.3 Kalimi nga XML në te Dhënat Tabelare 84
Kapitulli 9–SOA dhe bazat e të dhënave me kapacitet të lartë (Big Data) 86
9.1 Bazat e të dhënave me kapacitet të lartë 86
9.2 Problematikat në Bazat e të dhënave me kapacitet të Lartë (VLDB) 87
9.2.1 Koncepti i Particionimit 87
9.2.2 Përfitimi në Performancë me anë të Particionimit 89

9.2.3 Shërbimet Data Mining në të dhënat me kapacitet të lartë 91
9.2.4 SOA dhe të dhënat me kapacitet të lartë 92
9.3 Arkitektura "Të dhënat me kapacitet të lartë-si-shërbime" 93
Konkluzione 95
Referencat 97

Lista e Figurave

Figurë 1. Karakteristikat e SOA-s 1
Figurë 2. Komunikimi ndërmjet dy shërbimeve Web 2
Figurë 3. Pavarësia e një shërbimi 3
Figurë 4. Trasmetimi i mesazheve dhe proçesimi i tyre 4
Figurë 5. Arkitektura Enterprise Service bus 6
Figurë 6. Komponentet e arkitekturës së orientuar nga shërbimet 6
Figurë 7. Integrimi i Aplikimeve 8
Figurë 8. Mënyrat e Integrimit 11
Figurë 9. Shembull i 'Platform Integration' 11
Figurë 10. 'Data Integration' ndërmjet platformave të ndryshme 12
Figurë 11. Shembull i 'integrimit Komponent' 12
Figurë 12. Shembull i 'Application Integration' 13
Figurë 13. Komunikimi 'point-to-point' ndërmjet aplikimeve 17
Figurë 14. Aplikimet të lidhura nëpërmjet një 'integration hub' 17
Figurë 15. Skema 'File Transfer' 19
Figurë 16. Skema 'Shared Database' 20
Figurë 17. Skema 'Remote Procedure Invocation' 21
Figurë 18. Skema 'Message Bus' 21
Figurë 19. Koncepti i 'Web Service' 22
Figurë 20. Sistemet e bazave të dhënave të shpërndara 24
Figurë 21. Diagrami logjik për makina virtuale të bazuara në hypervisor 27
Figurë 22. Virtualizimi i shërbimeve 30
Figurë 23. Arkitektura Cloud Computing 31
Figurë 24. Koncepte të implementimit për Cloud Computing dhe SOA 33
Figurë 25. Migrimi Live i makinave Virtuale 36
Figurë 26. Paraqitja skematike per një datacenter 37
Figurë 27. Aplikacioni i bazuar në shërbimet SOA 40
Figurë 28. Të dhënat brenda dhe jashtë shërbimeve 44
Figurë 29. Shërbimet dhe mesazhet të lidhura bashkë 45
Figurë 30. Mesazhe të shumëfishta të ngjashme 46
Figurë 31. Shërbimet kundrejt komponenteve 46
Figurë 32. Integrimi i SQL Server 2005 me driver-in http.sys 48
Figurë 33. Nyjet http në SQL Server 2008 49

Figurë 34. Konvertimi nga CLR i CIL në kod native 50
Figurë 35. Arkitektura SQLCLR 51
Figurë 36. Topologjia dhe rrjedha DCN 52
Figurë 37. Një dialog i shërbimeve broker 56
Figurë 38. Grupet e Komunikimit 56
Figurë 39. Renditja e Mesazheve në SB 57
Figurë 40. Proçesi i besueshem i mesazheve në SB 57
Figurë 41. Objektet e shërbimit SB 58
Figurë 42. Struktura e shërbimit Broker dhe sinjali i dialogut 59
Figurë 43. Objektet SB të krijuara në SQL Server 2008 62
Figurë 44. Outputi i radhës Destinacion 64
Figurë 45. Disa prej aplikacioneve kryesore ne nje sistem bankar 65
Figurë 46. Sistemi audit i qendërzuar 67
Figurë 47. Konfigurimi i aplikacionit sipas mesazheve dhe radhëve 72
Figurë 48. Tabela audit pas aktiviteteve të ndryshme 73
Figurë 49. Krahasimi i performancës kur ka aktivitet paralel në të gjitha aplikacionet 75
Figurë 50. Integrimi i XMl në SQL Server 78
Figurë 51. Starndartet SQL/XML dhe XQuery 79
Figurë 52. Një DataSet në XML 79
Figurë 53. OPEN XML 84
Figurë 54. Proçedura SP XML Prepare Dokument 85
Figurë 55. Faktorët që ndikojnë në bazat e të dhënave me kapacitet të lartë 87
Figurë 56. Particionimi i tabelës Shitje per ç'do muaj te vitit 89
Figurë 57. Përfitimi në Performancë duke particionuar të dhënat 90
Figurë 58. SOA dhe të dhënat me kapacitet të lartë 94

Lista e Tabelave

Tabelë 1. Aplikacioni i Integruar në Makinën Virtuale 38
Tabelë 2. Krahasimi i kohës Live Migration për Hipervizorë të ndryshëm 38
Tabelë 3. Kohët e mbushjes së logut kur ka aktivitet në një aplikacion 73
Tabelë 4. Krahasimi i performancës duke përdorur shërbimet në brendësi të databazës 74
Tabelë 5. Kohët kur ka aktivitet paralel në të gjitha aplikacionet 74
Tabelë 6. Diferenca midis modeleve të dhënave Relacionale dhe të dhënave XML 77
Tabelë 7. Krahasime midis bazave të dhënave me kapacitet të lartë 86
Tabelë 8. Koha mesatare e egzekutimit te kërkeses për ç'do particion 90

Hyrje

Arkitekturat dominuese të aplikacioneve klient/server dhe shumë shtresore të dekadave të fundit kanë ngritur një numër të madh problemesh të cilat lidhen me përshkallëzimin dhe disponueshmërie e sistemeve IT. Të tjera probleme u shfaqën gjatë fazës për të inkorporuar sistemet e vjetra në ato të reja. Gjithashtu një problem i madh lidhej me faktin e ruajtjes së të dhënave në databaza shumë të mëdha, të qendërzuara ku të gjitha komponentet klient kishin akses direkt me databazën, kjo për faktin që për sisteme të tilla informacioni, rritej vështirësia e disponueshmërisë së tyre. Gjithashtu një ndër faktorët kryesore i cili cënohej në këto baza të dhënash ishte performanca e tyre apo e aplikacioneve që ndërvepronin me to, degradimi i të cilave lidhej direkt me rritjen e numrit të kërkesave nga aplikacionet apo rritja e të dhënave. Pas dekadash shperndarjeje të një numri shumë të madh sistemesh të ndërtuara në platforma dhe teknologji te ndryshme, bota ishte plotë me sisteme që e kryenin punën e tyre në mënyrë perfekte por që nuk kishin asnjë shteg të qartë për të ndërvepruar me aplikacione të tjera në një mjedis gjithmonë e më tepër të orientuar dhe të shpërndarë në infrastrukurën e rrjetit.

Arkitektura e orientuar nga shërbimet (SOA) është një model arkitekturor që udhëheq të gjithë aspektet e krijimit dhe përdorimit të proçeseve të biznesit të paketuara si shërbime. Kjo arkitekturë përcakton dhe siguron që infrastruktura IT të lejojë aplikime të ndryshme të shkëmbjenë të dhëna dhe të marrin pjesë në proçeset e biznesit pavarësisht nga sistemet e shfrytezimit apo gjuhët e programimit të këtyre aplikimeve.

SOA përfaqëson një model në të cilin funksionaliteti dekompozohet në njësi të vogla të veçanta (shërbime), të cilët mund të shpërndahen në rrjet dhe mund të kombinohen së bashku si dhe të ripërdoren për të krijuar aplikime biznes. Këto shërbime komunikojnë me njëri-tjetrin duke kaluar të dhënat nga një shërbim tek tjetri, ose duke koordinuar një aktivitet ndërmjet një ose më shumë shërbimesh.

Arkitekturat e bazave të dhënave të orientuara nga shërbimet (SODA) lidhet pikëritsh me implementimin e SOA në arkitekturën e bazave të dhënave ku secili komponent i saj në trajtën e shërbimit është i pavarur.Kjo mundëson që sistemet e mëdha të dekadës aktuale të shpërndara si Cloud,Big Data të cilat përdorin parimin e ndarjes në komponente shërbimesh më të vogla të jenë më rezistente ndaj dështimit dhe të ofrojnë një performancë shumë të lartë komunikimi.

Shkëmbimi i të dhënave ndërmjet aplikacioneve të ndryshme të cilët nuk janë të hapur(kompatibël) me njëri-tjetrin dhe që përdorin baza të dhënash të ndara kategorizohet si një operacion i komplikuar dhe me kosto për biznesin.

Në këtë punim do të trajtohet mënyrat e ndryshme që egzitojnë për integruar aplikimet e veçanta në organizata mbi sistemet e bazave të dhënave, në një bashkësi aplikimesh bashkëpunuese në mënyrë që zhvillimi i tyre të bëhet sa më fleksibël.

Do të trajtohet roli databazave si një nga faktorët kryesorë që luan rol në këtë integrim, ku do të përqendrohem në studimin e bazave e të dhënave të shpërndara

(distributed) dhe problematikat që shfaqen gjatë komunikimit.

Një nga avantazhet më të mëdha që ofron ky model shpërndarjeje është që pavarësisht se të dhënat janë te shpërndara fizikisht, në nivelin llogjik të dhënat mund të aksesohen nga kudo.

Qëllimi kryesor i punimit është që në një mjedis të shpërndarë bazash të dhënash të krijohet një shtresë ndërmjetëse e cila do të komunikojë me të gjithë databazat e shpërndara në rrjet duke ruajtur integritetin dhe performancën, pavarësisht nga specifikat apo numri i aplikacioneve heterogjene që ndodhen në këtë mjedis të shpërndarë.

Bashkësia këtyre të dhënave mund te jetë e shpërndarë fizikisht në vendndodhje të ndryshme të shpërndara ne rrjeta lokale, intranet-te apo me anë të internetit, midis të cilave shërbimet e bazave të dhënave komunikojnë me njëra-tjetrën.
Aksesimi i të dhënave lidhet direkt me komunikimin në rrjet,ku një nga pjesët e punimit do të jetë pikërisht studimi i performancës që ka rrjeti në ketë ambjent të shpërndarë dhe si mund të optimizohet përdorshmëria e rrjetit duke përzgjedhë protokollet e rrugëzimit të cilat ndikojnë në eleminimin e bllokimeve (starvation) në rrjet. Gjithashtu do të vlerësohet performanca që mund të arrihet ne rastet e përdorimit të protokollit IPv6 e duke përdorur teknikën “Flow Label”.

Shtresa ndërmjetëse (middleware) do të krijohet sipas arkitekturës së mesazheve dhe radhëve (Message Queuing) të cilat përdorin parimin asinkron dhe do të mundësojë që kërkesat për ndryshime në bazat e të dhënave të shpërndara relacionale apo aplikacioneve heterogjene, të implementohen direkt në nivel databaze duke ruajtur pavaresinë nga njëra-tjetra.

Që kjo shtresë ndërmjetese të komunikojë me bazat e të dhënave të ndryshme dhe të shpërndara, do ti ruajë të dhënat në standartin e njohur XML, i cili është i njohur dhe i integruar në shumicën e sistemeve të menaxhimit të databazave, të cilat suportojnë ruajtjen e të dhënave në format XML dhe gjenerimin e XML duke u nisur nga të dhëna relacionale.Kalimi nga formati XML në të dhëna relacionale dhe anasjelltas do të jetë pjesë e punimit.

Teknikisht në këtë punim do të përdoret ambjenti i sistemit të menaxhimit të bazave të dhënave Sql Server i cili suporton në brendësi arkitekturën e mesazheve dhe radhëve me anë të shërbimeve Broker të integruara në brendësi të tij.

Si rast studimi do të merret: krijimi i një sistemi auditi i qendërzuar në një bankë, i cili do të ruajë gjithë ndryshimet që ndodhin nga aplikacionet kryesore të përdorura në bankë që ndodhen në servera të ndryshëm në rrjet duke mos ndryshuar kodin në aplikacionet heterogjene të përdorura, por duke e mundësuar këtë nëpermjet shërbimeve të mesazheve dhe radhëve broker direkt në bazë të dhënash.

Kërkesat e dërguara nga bazat e të dhënave janë asinkrone, gjë kjo e cila do optimizojë ndjeshëm kohën e përgjigjes të ç’do aplikacioni që është pjesë e rastit të studimit.

Disa nga problematikat e trajtuara që do të trajtohen lidhur me këtë rast studimi janë:

- Rritja e log-ut audit në bazën e të dhënave qëndrore në mënyrë të pakontrolluar. Kur trendi i të dhënave në databazë rritet me kapacitete të larta, është vënë re një degradim i performancës.

  Si zgjidhje për këtë problem jam përqendruar në teknikën e arshivimit, dhe particionimit, duke përfituar në :
  mbajtjen nën kontroll të dhënave dhe duke përfituar në performancë.

- Qëndrueshmëria ndaj dëmtimit për bazën e të dhënave.
  Në sistemin e të dhënave të studimit i cili bashkëvepron me databazat e tjera në rrjet, të dhënat ruhen në makina virtuale në server. Një ndër kërkesat e domosdoshme është qëndrueshmëria ndaj dëmtimit për bazën e të dhënave.

  Meqënëse të dhënat janë me kapacitete të larta dhe ndryshimet për këtë rast janë të shpeshta, kam studiuar teknikën e Migrimit Live, e cila është dërgimi i shërbimeve dhe të dhënave nga një makinë fizike në një tjetër në kohë reale për të mundësuar qëndrueshmërinë ndaj dëmtimit.
  Faktor kryesor është performanca, dhe për këtë arsye në këtë rast jam përqëndruar në vazhdim në optimizimin e teknikës së Migrimit Live.

Ky sistem i qendërzuar dhe i madh do të shërbejë si burimi kryesor i teknikave data Mining që mundësohen në Sql Server me “Analysis Services”, të cilat në vazhdimësi do të përdoren për të krijuar modele të gatshme dhe parashikuese për auditorët e ndryshëm.

Punimi është i ndarë në nëntë kapituj.

Në kapitullin e parë trajtohet modeli i arkitekturës së orientuar nga shërbimet, përbërja infrastruktura, teknologjitë dhe zhvillimi i saj. Në këtë kapitull do të trajtohen komponentet e “Cloud Computing”, i cili mund të konsiderohet si një aplikacion i bazuar ne infrastrukturën SOA.

Në kapitullin e dytë punimi fokusohet në integrimin e aplikimeve që shpesh i referohemi dhe EAI (Enterprise Application Integration). Do të jepet nje paraqitje e kërkesave të tregut për integrimin dhe do të trajtohen llojet e ndryshme te integrimit ne nivel platfome, të dhënash, komponente dhe aplikacioni.

Në kapitullin e tretë trajtohen arkitekturat e integrimit të aplikimeve.
Një mënyrë shumë e rëndësishme për të bërë integrimin e aplikimeve të shkallëzueshëm është duke rritur numrin e hapave automatik dhe duke reduktuar numrin e hapave manual.Do të shpjegohen lidhjet midis aplikimeve dhe shtresat e ndërmjetme që shërbejnë per integrimin.

Në kapitullin e katërt diskutohet rëndësia dhe roli që ka infrastruktura, protokollet dhe komponentet e rrjetit mbi aplikacionet të cilat lidhen me baza të dhënash te shpërndara arkitekturën e orientuar ndaj shërbimeve.Do të krahasohen protokolle të cilat vlerësojnë gjendjen e rrjetit dhe të burimeve dhe duke u nisur nga kombinimi i elementeve te ndryshëm, optimizojnë rrugët dhe vonesat.

Në kapitullin e pestë trajtohet virtualizimi,virtualizimi i shërbimeve dhe lidhjet qe egzistojne midisa Virtualizimit, SOA dhe Cloud Computing. Do te trajtohet rëndësia e teknikës së Migrimit live cila lidhet me lëvizjen e të dhënave në makinat virtuale dhe performancën.

Në kapitullin e gjashtë do të studiohen bazat e të dhënave për mënyrën e integrimit te koncepteve SOA ne brendësi te databazes dhe mjetet qe e mundesojnë këtë, koncept i njohur ndryshe si Arkitekturat e Bazave të Dhënave të Orientuara nga Shërbimet (Soda).

Në kapitullin e shtatë do të ndërtohet një aplikacion middleware sipas arkitekturës së shërbimeve që dërgon të dhëna në formën e mesazheve dhe radhëve që përdorin parimin asinkron. Kjo do të bëhet e mundur në sistemin e menaxhimit te bazave të dhënave Sql Server me anë të shërbimeve Broker të integruara në brendësi të Sql Server.

Në kapitullin e tetë do të trajtohet standarti XML në bazat e të dhënave, i cili është i njohur dhe i integruar në shumicën e sistemeve të menaxhimit të databazave, të cilat suportojnë ruajtjen e të dhënave në format XML dhe gjenerimin e XML duke u nisur nga të dhëna relacionale.Kalimi nga formati XML në të dhëna relacionale dhe anasjelltas do të jetë pjesë e këtij kapitulli.

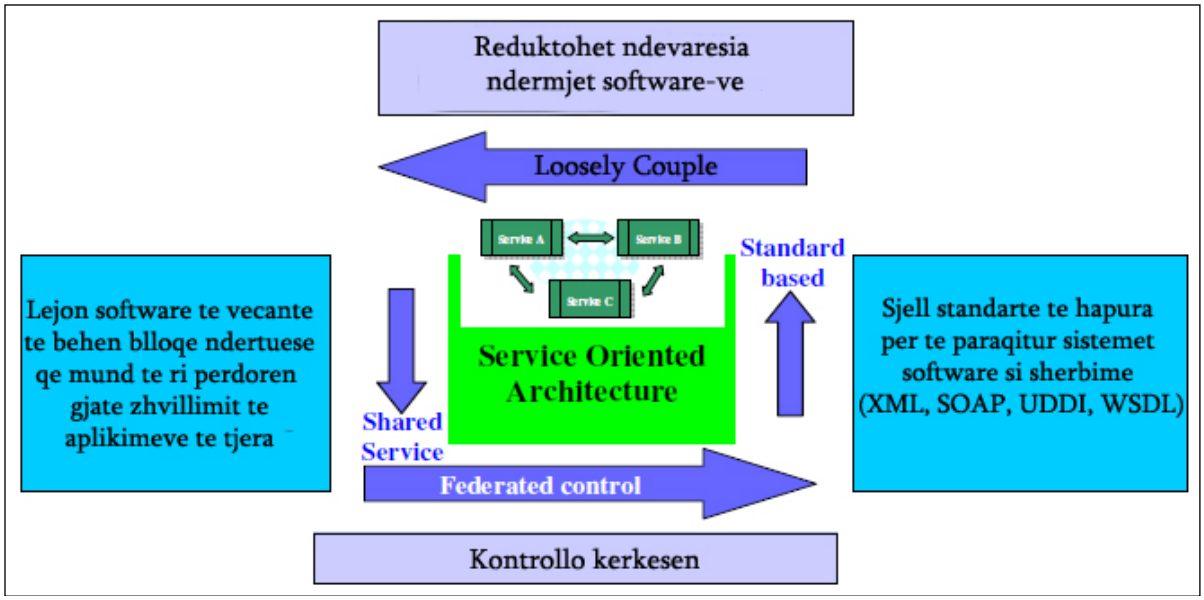
Në kapitullin e nëntë do te trajtohen ç'është jet që lidhen me bazat e të dhënave me kapacitet të lartë, performancën dhe krahasime të ndryshme, Sistemi e të dhënave të shpërndara,teknikat data mining dhe lidhja e arkitekturës së orientuar drejt shërbimeve me keto sisteme të ruajtjes së informacionit.

Punimi do të mbyllet me konkluzionet

Kapitulli 1 - Prezantim me Arkitekturën e Orientuar drejt Shërbimeve (SOA)

1.1 Arkitektura e Orientuar drejt Shërbimeve (SOA)

Arkitektura 'Service Oriented' (SOA) është një stil arkitekturor që udhëheq të gjithë aspektet e krijimit dhe përdorimit të proçeseve të biznesit të paketuara si shërbime. Gjithashtu SOA përcakton dhe siguron që infrastruktura IT të lejojë aplikime të ndryshme të shkëmbjenë të dhëna dhe të marrin pjesë në proçeset e biznesit pavarësisht nga sistemet operative ose gjuhët e programimit të këtyre aplikimeve.[1]SOA përfaqëson një model në të cilin funksionaliteti dekompozohet në njësi të vogla të veçanta (shërbime), të cilët mund të shpërndahen në rrjet dhe mund të kombinohen së bashku si dhe të ripërdoren për të krijuar aplikime biznes. Këto shërbime komunikojnë me njëri-tjetrin duke kaluar të dhënat nga një shërbim tek tjetri, ose duke koordinuar një aktivitet ndërmjet një ose më shumë shërbimesh. Figura e mëposhtme përshkruan disa nga karakteristikat e arkitekturës SOA.



Figurë 1. Karakteristikat e SOA-s

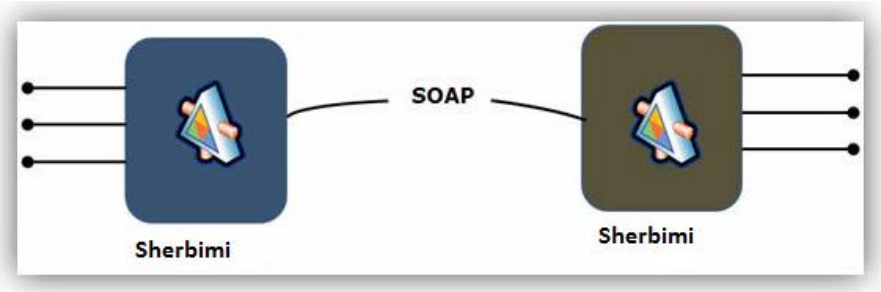
1.1.1 Përbërja, Ndërtimi, Infrastruktura

Duhet që aftësitë e software-ve të shpërndahen dhe të përdoren. Sigurisht që këto shërbime mund të implementohen si një bashkësi sistemesh. Këtë mundësi e jep arkitektura SOA, e cila na jep mundësinë e shpërndarjes, manaxhimit dhe përdorimit të këtyre burimeve. Me zhvillimin e teknologjisë gjithçka ka evoluar duke kaluar nga modulet tek objektet, tek komponentet dhe tani tek shërbimet.

SOA në mënyrën më të mirë lejon përhapjen e një gjenerate të re të aplikacioneve dinamike. Këto aplikacione dinamike janë ato që ju duhen sot bizneseve për të automatizuar veprimet manuale, me qëllim orkestrimin e proçeseve të biznesit. Orientimi drejt shërbimeve jep mundësinë e integrimit të burimeve të ndryshme në sisteme të ndryshme. Në qoftëse do të shtrojmë pyetjen se çfarë është SOA, si përgjigje vijnë dy përkufizime të ndryshme, ndoshta edhe konfliktuale.
Disa përkufizime e sjellin SOA si një infrastrukture IT për të rritur efiçensën e biznesit, kurse disa të tjera e përkufizojnë ate si një mënyrë për të rritur efiçensën e IT.

1.1.2 Zhvillimi i SOA

[1]Shërbimet janë një zhvillim i teknologjisë së viteve 80 dhe 90. Ato, në vitet 80 njiheshin si objektet e orientuara nga modelet, më pas si komponente, dhe tani njihen si shërbime të orientuara nga modelet. Shërbimet e orientuara i trashëgojnë të gjithë karakteristikat e dy modeleve paraardhëse (si vetë përshkrimi, enkapsulimi, natyra dinamike), por ajo që është më e rëndësishme është zhvillimi që erdhi më pas - si transmetimi i mesazheve ndërmjet shërbimeve. Skema e mëposhtme e ilustron më së miri një shembull tipik komunikimi ndërmjet dy web serviseve.



Figurë 2. Komunikimi ndërmjet dy shërbimeve Web

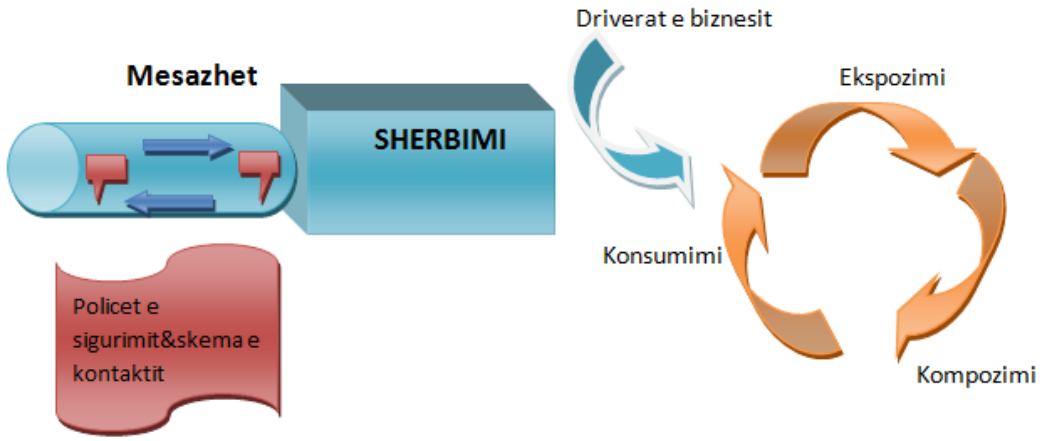
Pjesa kryesore tek shërbimet është pa diskutim vetë shërbimi. Shërbimi në vetvete është një program, i cili mund të instalohet në dy pjesët që do të komunikojnë. Gjithashtu shërbimi mund të përkufizohet edhe si një proçes specifik i një sistemi operimi si Unix ose Windows. Çelësi në këto shërbime mbetet arkitektura me përdorimin e një bashkësie kompjuterash të shpërndarë. Përdorimi i kësaj arkitekture na lejon publikimin e shërbimeve, që janë burimet, dhe jo adresimin tek implementimi i tyre. Me anë të SOA, funksionalitetet e aplikacioneve ekspozohen nëpërmjet një koleksioni shërbimesh. Këto shërbime janë të pavarura dhe përfshijnë llogjikën e biznesit dhe të dhënat. Kjo mënyrë bën që ndryshimet që ndodhin tek ofruesi i shërbimit të mos ndikojnë tek konsumatori i shërbimit. Duke mbajtur një ndërfaqe konsistente konsumatori i shërbimit mund të zgjedhë një instancë alternative të të njëjtit shërbim pa modifikuar kërkesat e aplikacionit. Ofruesi i shërbimit dhe konsumatori nuk janë të detyruar të përdorin të njëjtën teknologji. Avantazhi i kësaj mënyre është se bën të mundur që programe të shkruara në gjuhë të ndryshme, në

[1] http://msdn.microsoft.com/en-us/library/bb833022.aspx

platforma të ndryshme, të komunikojnë me njëra-tjetrën duke përdorur një mënyrë standarte.

Anatomia e një Web Service

Shërbimet e paraqitura si në figurën 3, përdoren gjerësisht për të ekspozuar investimet në IT. Shërbimet vetë mund të kompozohen në proçese biznesi, për të qenë të përdorshme më pas nga përdoruesit, sistemet apo edhe nga shërbimet e tjera.



Figurë 3. Pavarësia e një shërbimi

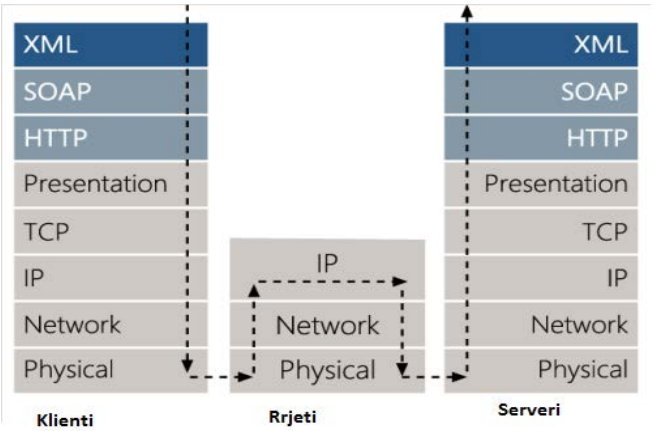
Në këtë model, kryesore është ndarja e ndërfaqes nga implementimi. Konsumatori i shërbimit duhet dhe ka nevojë të kuptojë vetëm ndërfaqen. Implementimi ndodh gjatë kohës pa e shqetësuar klientin. Shumë përfitime që i adresohen sot SOA, vijnë nga ky abstragim. Kjo do të thotë që e njëjta ndërfaqe mund të aksesohet nga shumë klientë njëkohësisht. Ofruesit e shërbimeve deklarojnë një kontratë për të paraqitur ndërfaqen publike që ata ofrojnë. Të gjitha lidhjet me ofruesin e shërbimit ndodhin nëpërmjet ndërfaqes. Kjo ndërfaqe përmban proçese publike si edhe të dhëna. Proçeset publike përbëjnë pikën hyrëse në shërbim, kurse të dhënat prezantojnë mesazhin që do t'i dërgohet shërbimit. Në qoftëse do të përdorim një dokument WSDL për të paraqitur një kontratë të thjeshtë, atëherë "mesazhi" prezanton të dhënat. Ndërkohë që pjesa më e vështirë e gjithë kësaj është ekspozimi i ndërfaqes publike dhe mirëmbajtja e saj.

1.1.3 Karakteristikat kryesore të SOA.

Tre konceptet kyesore të SOA janë:
Ekspozimi: fokusohet tek sesi shërbimet do të ekspozohen, në mënyrë që të jenë të përdorshme nga klientët.
Kompozimi: fokusohet në kombinimin e shërbimeve në aplikacione, apo proçese biznesi.

Konsumi: fokusohet në mënyrën se si këto shërbime të kompozuara do të përdoren nga klientët, përdoruesit fundorë.

Në shumicën e rasteve SOA mund të implementohet duke përdorur disa standarte, ku më të përdorurat janë HTTP, WSDL (Web Services Definition Language), SOAP (Simple Object Access Protocol) dhe XML (eXtensible Markup Language). Këto standarte punojnë së bashku për të transmetuar mesazhet nga një shërbim tek një tjetër. Në qoftëse në dërgojmë një letër, atëherë në duhet ta dimë adresën se ku do ta dërgojmë atë. Në rastin e SOA ky informacion vjen nga WSDL e cila liston mundësitë e trasnmetimit si (HTTP, FTP, SMTP) si edhe gjuhën standarte të komunikimit që është XML. Në të njëjtën mënyrë zarfi SOAP mban informacion rreth adresave ku do të dërgohen mesazhet dhe se kush e ka autorizimin t'i hapë ato. Në zarfin SOAP nuk përcaktohet se kush do ta marrë zarfin por emri i makinës remote që do ta proçesojë informacionin që ndodhet brenda tij. Me këtë informacion ne mund të krjojmë një mesazh të bazuar në standartin XML, ta fusim brenda një zarfi dhe ta nisim nëpërmjet protokollit HTTP. Marrësi i mesazhit lexon zarfin dhe përcakton makinën remote, adresën e saktë. Mesazhi më pas kalon në këtë makinë remote për proçesim të mëtejshëm. Ajo që në shohim në këtë proçes është kalimi i mesazhit në shumë hapa që nga përgatitja deri tek mbërritja në destinacion. Kjo përbën edhe një nga sfidat e mëdha të SOA, trasmetimin e mesazheve dhe proçesimin e tyre.



Figurë 4. Trasmetimi i mesazheve dhe proçesimi i tyre

1. **Shërbimi** është një koncept shumë i rëndësishëm. Web shërbimet janë teknologjitë që realizojnë shërbimet, nëpërmjet të cilave ato mund të publikohen, zbulohen, dhe përdoren.
2. **SOA** në vetvete nuk është vetëm një arkitekturë shërbimesh, pasi ajo përfshin politikat, kontratat si edhe ndërfaqet nëpërmjet të cilave kontrollohet konsumi i shërbimeve.

SOA: Është një arkitekturë e bazuar në një bashkësi kompjuterash të shpërndarë (loosely-coupled architecture) e dizenjuar për të ndihmuar bizneset të arrijnë objektivat e tyre.
SOA: Konsiderohet si një bashkësi komponentesh të cilët mund të shpërndahen, dhe ndërfaqet e tyre përshkruese mund të publikohen dhe të zbulohen.
Për SOA ekzistojnë tre tipe arkitekturash:

1. **Arkitektura e aplikimit**: Kjo është ndërfaqja që u ofrohet bizneseve, të cilët konsumojnë shërbime nga një ose më shumë ofrues shërbimesh duke i integruar ato në proçeset e biznesit.

2. **Arkitektura e shërbimit**: Kjo shërben si një urë ndërmjet implementimit dhe konsumit të aplikacioneve, duke krijuar në këtë mënyrë një pamje llogjike me një bashkësi shërbimesh të aksesueshme.

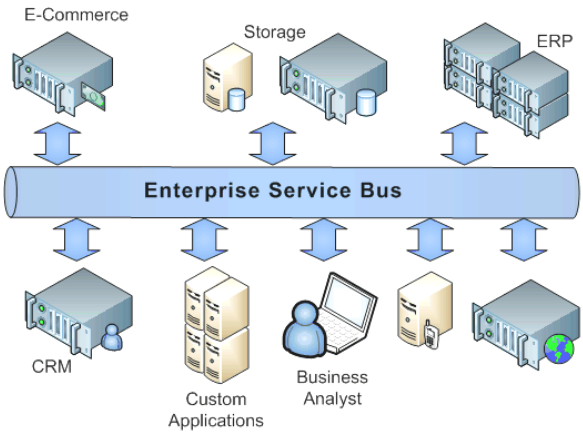
3. **Komponentet e arkitekturës**: Kjo përshkruan një mjedis të mundshëm që mbështet implementimin e aplikacioneve, objektet e biznesit dhe implementimin e tyre.

Këto arkitektura mund të studiohen nga këndvështrimi i konsumuesit si edhe i ofruesit të shërbimit. Çelësi i këtyre arkitekturave është se konsumuesi i shërbimit nuk interesohet për detajet e imlementimit të shërbimit. Konsumatori fokusohet tek arkitektura e aplikimit, përdorimi i shërbimit, por jo tek komponentet e arkitekturës dhe implementimi i databazës. Ndërkohë që ofruesi i shërbimit fokusohet tek komponentet e arkitekturës, arkitektura e shërbimit, por jo në arkitekturën e aplikimit. Por, si konsumatori i shërbimit ashtu edhe ofruesi kanë nevojë të dinë gjithë informacionin rreth aplikacioneve.

Në thelb të kësaj arkitekture qëndron manaxhimi i shërbimeve dhe shpërndarja e tyre sipas një rradhë. Pasi është shërbimi, ai që lidh konsumuesin me ofruesin. Që konsumuesi të dijë se për çfarë shërbimi ai ka nevojë, cilat janë ato shërbime që janë të lira për t'u aksesuar, apo se kush janë ato shërbime që operojnë së bashku, u zhvillua koncepti i Business Service Bus (BSB). BSB është një pamje llogjike që prezanton të gjitha shërbimet që janë të përdorshme për një domain të caktuar.

[2] Për të integruar sistemet IT egzistuese me ato të reja, një arkitekturë e orientuar drejt shërbimeve i duhet një infrasktrukturë e cila mund lidhë me njëri-tjetrin ç'do burim IT, pavarësisht teknologjisë apo ku është i zhvilluar.Një infrastrukturë e tillë është dhe Enterprise Service Bus (ESB) e paraqitur si në figurën 5.

[2] http://www.centeractive.com/content/enterprise-service-bus



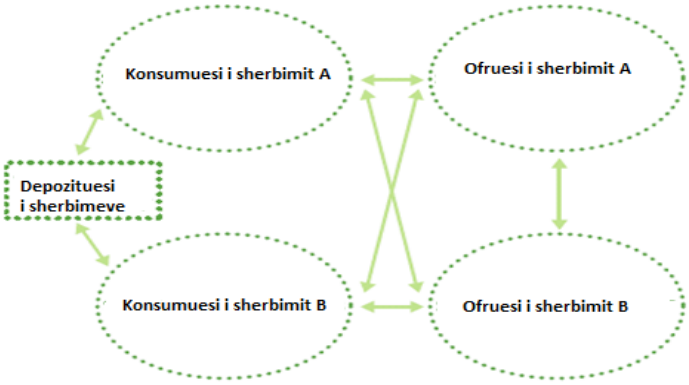
Figurë 5. Arkitektura Enterprise Service bus

1.2 Komponentet arkitekturore të SOA

Komponentet bazë të një arkitekture të orientuar nga shërbimet janë:

1. **Ofruesi i shërbimeve** (Sevice providers): Një ofrues shërbimesh është një komponente ose një bashkësi komponentesh që ekzekuton funksionet e biznesit, gjithashtu pranon inpute dhe outpute.
2. **Konsumuesi i shërbimeve** (Service consumer): Një konsumues shërbimesh është një bashkësi komponentesh të interesuar të përdorin shërbimet që mundësohen prej ofruesve të shërbimeve.
3. **Depozituesi i shërbimeve** (Service repository): Një depozitues përmban përshkrimet e nevojshme për të gjitha shërbimet që ofrohen nga shërbyesit e ndryshëm. Ofruesit e shërbimeve i depozitojnë shërbimet që ata ofrojnë në këto komponente dhe konsumuesit e tyre mund të aksesojnë ato kur ata kanë nevojë.

   Këto komponente bazë janë paraqitur në figurën 6.



Figurë 6. Komponentet e arkitekturës së orientuar nga shërbimet

1.3 Avantazhet, Sfidat.

1.3.1 Avantazhet e SOA.

Avantazhet e SOA shihen në dy këndvështrime: në IT dhe në biznes.

1. Duke e parë nga pozicioni i IT, ajo lehtëson manaxhimin e burimeve të shumta të shpërndara përmes shumë platformave, të cilat kërkojnë më pak investime në hardware dhe janë më të besueshme e më pak të kushtueshme. Kurse, duke e parë SOA nga pikëpamja e biznesit, ajo lejon zhvillimin e një gjenerate të re aplikacionesh dinamike që janë zgjidhja e shumë problemeve të biznesit.

2. SOA krijon lidhje të forta midis klientëve dhe ofruesve të shërbimeve. Duke i bërë aplikacionet dinamike dhe shërbimet e bizneseve të përdorshme për përdoruesit e jashtëm dhe ofruesit, ajo arrin jo vetëm një bashkëpunim më të mirë, por rrit në këtë mënyrë edhe kënaqësinë e bashkëveprimit.

3. SOA duke ndihmuar në rritjen e nivelit të informimit për aplikacionet, dhe rritjen e aksesueshmërisë së shërbimeve të biznesit arrin në këtë mënyrë që të dyja palët t'i kuptojnë më mirë kërkesat që kanë ndaj njëri-tjetrit.

Burimet IT, të shpërndara dhe komplekse përbëjnë thelbin e bizneseve. Por, këto burime vite më parë nuk përfaqësonin atë që priste biznesi prej tyre. Zgjidhja e këtij problemi nuk është zëvendësimi i aplikacioneve apo sistemeve, por përdorimi i shërbimeve të SOA si urë për të plotësuar nevojat e biznesit. Orientimi drejt shërbimeve e arrin këtë qëllim, duke i bërë sistemet më të afërta ndaj kërkesave të biznesit. Ndjekja e kësaj linje bën që investimet në IT të përshtaten me strategjitë e biznesit.

1.3.2 Sfidat e SOA.

Cilat janë sfidat që i paraqiten SOA-s.

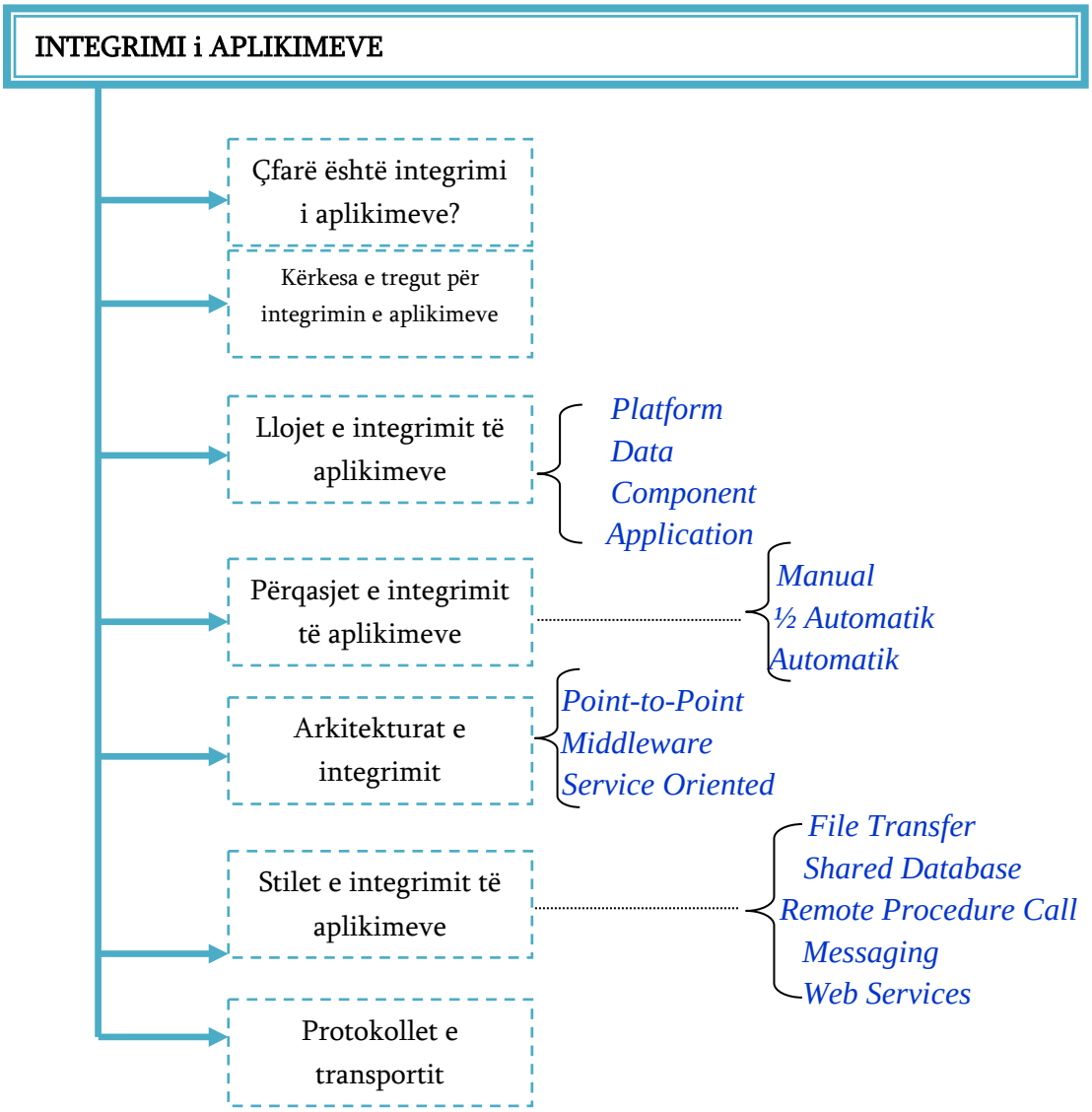
[42]Gjithmonë zhvillimet kanë edhe mangësitë e tyre. Edhe tek SOA të gjitha këto zhvillime shoqërohen nga ana tjetër edhe me disa mangësi.

Një prej tyre është edhe siguria e cila është shumë e rëndësishme në fushën e teknologjisë, mungesa e së cilës sjell vjedhjen e të dhënave sensitive.Prandaj është i nevojshëm validimi të dhënave dhe rritja e masave të sigurisë. Marrja e masave të nevojshme ndalon sulmet e shumta dhe rrit performancën duke reduktuar duplikimin e kodit. Rritja eksponenciale e numrit të lidhjeve shkakton uljen e shpjetësisë së komunikimit. Menaxhimi i numrit të madh të lidhjeve do të reduktojë numrin e serverave, uljen e kostos, reduktimin e kompleksitetit dhe rritjen e performancës. Përdorimi i memorjes kashe dhe hostimi i shërbimeve të përbashkëta është një mundësi që duhet aplikuar.

Rritja e madhësisë së mesazheve SOAP. Kjo kërkon më shumë bandwith dhe shumë burime për t'u konsumuar. Zgjerueshmëria (scalability), standarti XML që përdoret për të transmetuar mesazhet kërkon një infrastrukturë më të përshkallëzueshme. Por nga ana tjetër kjo sfidë sjell edhe rritjen e performancës.

Kapitulli 2 . Integrimi i Aplikimeve

Ky kapitull do të trajtojë në mënyrë të detajuar integrimin e aplikimeve, ku struktura e organizimit paraqitet në figurën e mëposhtme.



Figurë 7. Integrimi i Aplikimeve

2.1 Çfarë është Integrimi i Aplikimeve?

Integrimi i aplikimeve që shpesh i referohemi dhe EAI (Enterprise Application Integration) është një term kompjuterik biznesi për planet, metodat dhe mjetet që synojnë modernizimin, konsolidimin dhe koordinimin e aplikimeve kompjuterike në një korporatë.[2] Zakonisht, një korporatë ka aplikime dhe baza të dhënash ekzistuese të trashëguara dhe synon të vazhdojë t'i përdor këto sisteme ndërkohë që shton apo migron në teknologji të reja.[3]

Integrimi i aplikimeve është ndarja dhe përdorimi i përbashkët, në mënyrë të sigurtë dhe të orkestruar, i të dhënave dhe proçeseve ndërmjet aplikimeve të ndryshme të një kompanie.[4]

2.2 Nevoja dhe kërkesa e tregut për Integrimin e Aplikimeve

Me kalimin e viteve, mjediset IT janë bërë gjithnjë e më shumë komplekse dhe heterogjene për shkak të nevojave të ndryshme të klientit dhe risive të menjëhershme në industrinë e IT.[5] Mjediset moderne sipërmarrëse përbëhen nga një numër i madh aplikimesh, që janë zhvilluar ndër vite, sa herë që ka pasur dhe janë idenifikuar kërkesa të reja nga klienti/biznesi. Për rrjedhojë, këto aplikime në më të shumtën e rasteve, janë të periudhave të ndryshme, janë shkruar nga persona të ndryshëm duke përdorur gjuhë dhe teknologji të ndryshme, qëndrojnë në platforma hardware të ndryshme, përdorin sisteme operative të ndryshme, si dhe ofrojnë funksionalitete të ndryshme. Për kompaninë, këto kushte mund të cojnë në aktivitete të tepërta, kosto më të larta si dhe në përgjigje jo efiçente ndaj klientit. Ashtu sic punonjësit duhet të punojnë së bashku për të realizuar qëllimet e biznesit, edhe aplikimet kanë nevojë të bëjnë të njëjtën gjë.[6]

Në mungesë të integrimit të aplikacioneve, kompanitë nuk do të jenë në gjendje të sistemojne (leverage) informacionin prezent në depot (repositories) 'në zgjerim të vazhdueshëm' të ndërmarrjes dhe mund të rrezikojnë të marrin vendime jo korrekte për shkak të informacionit kontradiktor (inconsistent). Një ndërmarrje (enterprise) funksionon në mënyrën më të mirë kur sistemon informacionin që është prezent në depo të ndryshme në ndërmarrje.[7] Mqs shumica e aplikacioneve prezentë në ndërmarrje nuk janë 'të hapur (open)' ose 'kompatibël' me njëri-tjetrin, atëherë kërkohet përpjekje shtesë për të integruar aplikacionet e ndryshme brenda ndërmarrjes.

Disa nga aplikimet e trashëguara janë kritike për realizimin e funksioneve të biznesit dhe për rrjedhojë nuk mund të zëvëndësohen. Një pjesë e këtyre aplikimeve kanë nevojë të integrohen me sisteme të tjera. Të tjerë mund të kenë nevojë të migrohen në një arkitekturë të re ose të ri implementohen duke përdorur teknologji të tjera. Sfida që shumë kompani tashmë po ndeshin është se si të sigurojnë një metodë nëprëmjet së cilës këto aplikime të mund të punojnë së bashku për të adresuar kërkesat e biznesit që vazhdimisht evoluojnë.[8]

Integrimi i sistemeve kompjuterike me njëri tjetrin erdhi si përgjigje ndaj nevojës për të lidhur së bashku shumë sisteme të veçanta kompjuterike. Sistemet e integruara supozohet të bashkëpunojnë dhe të sigurojnë funksionalitete të unifikuara. Për më tepër, pritet që një sistem i ri i integruar të operojë mbi të dhënat e siguruara nga të gjithë sistemet kompjuterike pjesëmarrëse. Ka shumë faktorë që e bëjnë integrimin të vështirë dhe sfidues. Sistemet, që supozohet se duhet të integrohen, mund të operojnë në organizata të ndryshme, mund të jenë zhvilluar duke përdorur teknologji të ndryshme, mund të jenë sisteme të trashëguara pa mirëmbajtës etj.

Nevoja për të integruar sistemet mund të lindë nga shumë arsye, psh, bashkimi i dy kompanive, bashkimi i shumë degëve të një kompanie, nevoja për të pasur një sistem përgjegjës për koordinimin e të tjerëve. Një arsye tjetër mund të jetë reduktimi i kostos. Një sistem i unifikuar, nga këndvështrimi i biznesit, është më efektiv dhe më i lirë të mirëmbahet se sa dy sisteme që punojnë në mënyrë të pavarur. Rezultate të pritshme nga integrimi janë si më poshtë:[9]

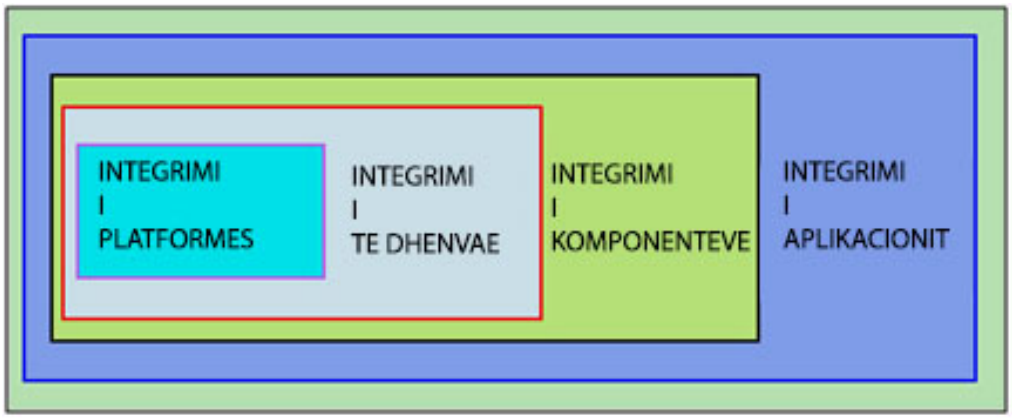
- Reduktohet kosto
- Rritet efikasiteti
- Përmirësohen proçeset e bisnesit (business processes)
- Unifikohet rrjedha e informacionit (flow of information)
- Arsye të tjera që e bëjnë proçesin e integrimit akoma më të vështirë janë:
- Sistemet që duhet të integrohen, mund të jenë të shpërndarë gjeografikisht në makina të vendosura në vendndodhje të largëta.
- Sistemet mund të jenë shkruar në gjuhë të ndryshme programuese
- Sistemet mund të mos kenë dokumentacion e nevojshëm teknik
- Sistemet mund të ekzekutohen në mjedise të ndryshme (sisteme operative, konfigurimet hardware etj)
- Sistemet mund të menaxhohen nga njësi të ndryshme organizative ose edhe kompani të ndryshme
- Kompanitë, që sigurojnë proprietary software, nuk janë të gatshëm të marrin pjesë në proçesin e integrimit, sepse nuk duan të zbulojnë implementimin e brendshëm të produkteve të tyre.
- Punonjësit mund të mos jenë të gatshëm t'u përshtaten ndryshimeve si rrjedhojë e proçesit të integrimit.

Realizimi në mënyrë efiçente i integrimit të aplikimeve i siguron çdo kompanie përfitimet e mëposhtme:

1. Aplikimet funksionojnë në mënyrë më efiçente dhe me kosto më të ulët.
2. Lejon modifikimin e proçeseve të biznesit sipas kërkesave të kompanisë
3. Siguron më shumë kanale shpërndarjeje për kompaninë
4. Lejon të shtohen hapa automatikë në proçese biznesi që më parë kërkonin ndërhyrje manuale.

2.3 Mënyrat e Integrimit të Aplikimeve

Ekzistojnë sisteme të ndryshme për të ndarë/shkallëzuar zgjidhjet e integrimit në nivele të ndryshme. Një metodë është të ndahen teknikat e integrimit në: integrim i brendshëm dhe integrim i jashtëm. Një kategorizim më i detajuar fokusohet në fleksibilitetin dhe shkallëzueshmërinë e zgjidhjeve të integrimit. Në figurën 8 më poshtë, janë paraqitur katër mënyrat e integrimit Integrimi 'Platform', 'Data', 'Component' dhe 'Application'[10].

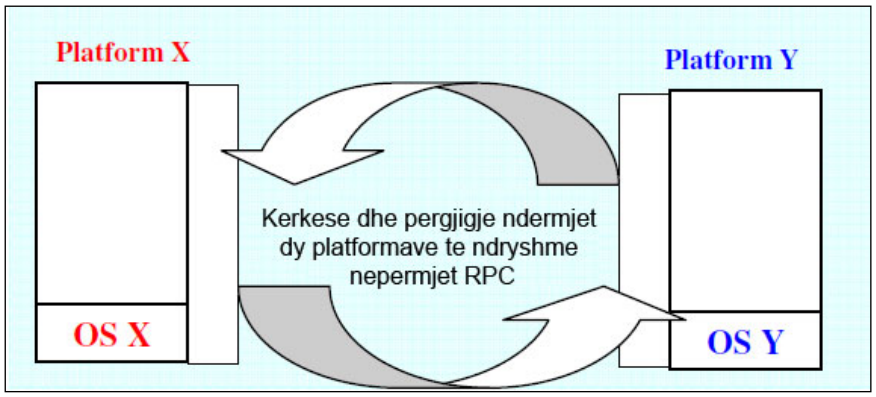


Figurë 8. Mënyrat e Integrimit

Në vijim do të paraqitet një përshkrimi i shkurtër i katër llojeve të integrimit si më sipër.

2.3.1. 'Platform Integration'

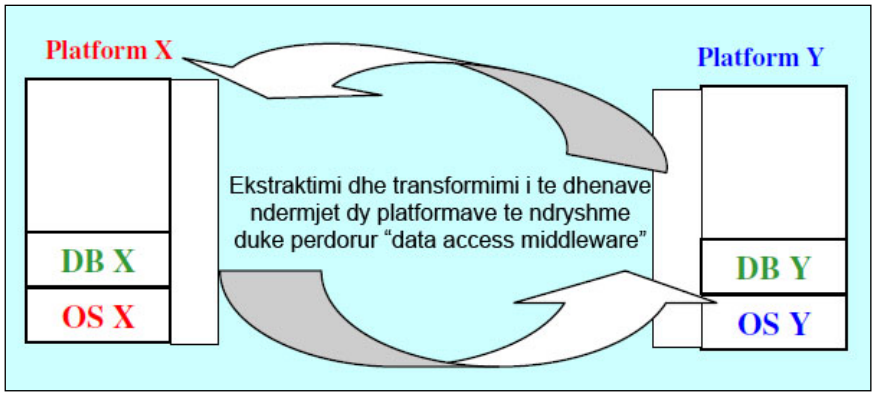
Integrimi 'Platform' siguron lidhjet dhe ndërfaqet e shërbimeve ndërmjet sistemeve me hardware, sisteme operative dhe aplikime të ndryshme. Zgjidhjet janë individuale dhe lidhja bëhet nëpërmjet 'Borkers' ose 'RPC' (Remote Procedure Calls) etj.[11] Duke qenë se secila zgjidhje integrimi është individuale, ngarkesa (workload) është po aq e lartë sa integrimi fillestar kur sisteme të reja duhet të shtohen në arkitekturën aktuale. Figura e mëposhtme tregon një shembull të arkitekturës së integrimit të tipit 'platformë' ndërmjet dy platformave të ndryshme.



Figurë 9. Shembull i 'Platform Integration'

2.3.3 'Data Integration'

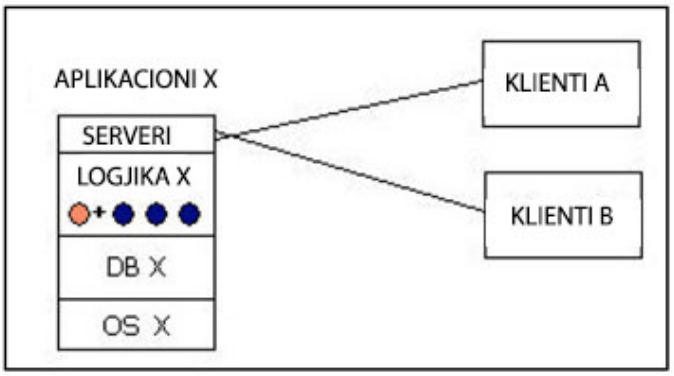
Zgjidhjet 'Data Integration' ofrojnë mjete për të ekstraktuar, transformuar dhe zhvendosur të dhëna.**[12]** Përfshin integrimin 'platformë' dhe kërkon informacion mbi skemat e databaza-ve. Një shembull i zakonshëm i 'data integration' është kur të dhënat nga një databazë shtohen në një databazë tjetër nëpërmjet SQL. Në shumë raste, të dhënat mund të integrohen ndërmjet dy platformave dhe sistemeve database të ndryshme, gjë që do të kërkojë përdorimin e 'API' dhe 'database connectivity drivers' për të aksesuar secilin nga 'database servers' në mënyrë që të ngarkohen të dhënat në një tjetër. Kjo tregohet në figurën si më poshtë:



Figurë 10. 'Data Integration' ndërmjet platformave të ndryshme

2.3.4 'Component Integration'

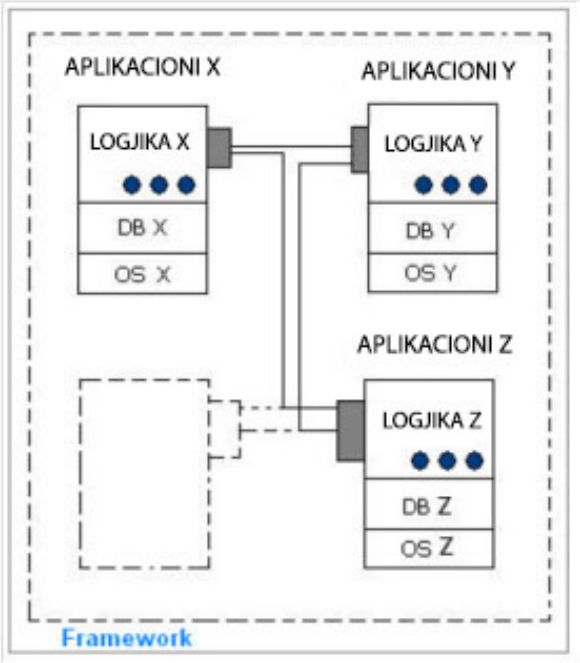
Zhvillimi i një 'data integration' mund të konsiderohet si 'component integration'. Tipare të shumta rrjeti i shtohen produktit si psh: 'load balance', 'fault protection', 'session management' dhe 'security'. Baza e një zgjidhje 'component integration' është një server që ofron të gjithë karakteristikat e mësipërme. Figura e mëposhtëme përfaqëson një shembull të 'component integration'.



Figurë 11. Shembull i 'integrimit Komponent'

2.3.2 'Application Integration'

Zgjidhjet ‘application integration’ ofrojnë një ‘framework’ për krijimin dhe ndryshimin e zgjidhjes së integrimit si psh shtimi i aplikimeve të reja shpejt dhe me lehtësi. Framework-u konsiston në ‘pre-built adapters’ për sistemet më të zakonshme të cilat zvogëlojnë kohën për shtimin e aplikimeve të reja.[13] Madje disa lidhin secilin aplikim me një formë të unifikuar. Ky trajtim ofron një avantazh të madh kur aplikimet ndryshojnë duke qenë se vetëm lidhja (mapping) me formën e unifikuar duhet të ndryshohet, të gjithë aplikimet e tjera janë të paprekur. Një zgjidhje ‘application integration’ përfshin të gjithë shërbimet e përshkruara në seksionet pasuese të ketij kapitulli. Figura e mëposhtme tregon si mund të integrohen aplikime të ndryshme së bashku dhe fleksibilitetin e ndryshimeve në topologjinë e integrimit.



Figurë 12. Shembull i ‘Application Integration’

‘Framework’ i para-zhvilluar e bën të lehtë shtimin e aplikimeve të reja, duke përdorur lloje të ndryshme përshtatësish

Qëllimi i integrimit të aplikimit është të krijohet një ‘framework’ për të integruar sisteme jo kompatibël dhe të shpërndarë, duke e bërë më të shpejtë dhe më të lehtë zgjerimin e proçeseve të biznesit përgjatë organizatës. Sot, shumë kompani kryesore përpiqen dhe tentojnë të zhvillojnë ‘frameworks’, mjete integrimi dhe infrastrukture ‘service-oriented’ në mënyrë që të realizojnë këtë detyrë. Mund të konstatohet se përpjekjet e pasukseshme kundrejt interoperabilitetit të aplikimeve brenda organizatës kanë sjell keqinterpretim të konceptit të integrimit të aplikimeve nga shumë organizata. Një nga pyetjet që shtrohet si rrjedhojë është:

“Çfare është ‘enterprise application integration’? Çfarë impakti ka kjo teknologji mbi ndërveprimin e aplikimeve dhe si mund të përfshihet në një organizatë?

Përpara se sa t'i përgjigjemi këtyre pyetjeve, në duhet të japim një përcaktim për termin 'enterprise integration' – "detyra që bën aplikime të ndryshme të punojnë së bashku për të prodhuar një tërësi funksionalitetesh të unifikuara".[12] Kjo detyrë është më shumë se vetëm thjesht integrimi i aplikimeve por duhet të marrë në konsiderate kriteret, opsionet e integrimit, stilet, politikat dhe gjithashtu përcaktimi i përqasjes më të mirë për integrimin e aplikimeve.

Integrim i aplikimeve është një pjesë shumë e rëndësishme e strategjisë IT të çdo biznesi. Bizneset po mendojnë se si të kombinojnë aplikimet e tyre të veçanta në një bashkësi aplikimesh bashkëpunuese në mënyrë që biznesi i tyre të bëhet më fleksibël. Një biznes fleksibël kërkon detyrimisht IT fleksibël.

Integrimi manual i aplikimeve

Integrimi manual i aplikimeve është proçesi kur personeli (psh. punonjesit ose klientet) duhet të vëprojë si ndërfaqe ndërmjet aplikimeve dhe të lehtësojë integrimin mes tyre.[4] Kjo mund të jetë në formën e mbledhjes së informacionit dhe regjistrimit të informacionit në disa sisteme dhe rrjedhimisht duke lexuar informacion nga këto sisteme në mënyrë që t'i përgjigjemi kërkesave të klientit. Në raste të tjera, një person mund të nevojitet të lexojë informacionin mbi klientin nga një databazë, dhe më pas ta ri-regjistrojë atë në një databazë tjetër që përdoret për qëllime të tjera. Kjo formë integrimi kërkon shumë pak investim në teknologji por më shumë përpjekje nga përdoruesi dhe zhvilluesit. Disavantazhet e identifikuara me teknikën manuale janë:

- Në më të shumtën e rasteve ka një probabilitet të lartë të pasaktësive në të dhëna për shkak të inkonsistencës dhe mungesës së shkallëzueshmërisë.
- Integrimi manual bëhet më kompleks kur ka më shumë sisteme që varen nga të dhënat dhe gjithashtu kur organizata bëhet më komplekse.
- Me rritjen e sasisë dhe kompleksitetit të të dhënave ose me rritjen e numrit të aplikimeve, organizata do të kërkojë më shumë njerëz për të mirëmbajtur një mjedis të tillë manual integrimi.

Një mjedis që mbështetet shumë në integrimin manual është zakonisht shumë jo efiçent, dhe nuk zgjerohet aq lehtësisht si mjediset që përdorin teknika më të automatizuara.

Integrimi ½ automatik i aplikimeve

Nga ana tjetër, integrimi ½ automatik kombinon hapat njerëzore me pak automatizim. Njeriu nuk mund të përfshihet në një fushë ku zgjidhja automatike korresponduese është shumë e vështirë ose e shtrenjtë për t'u implementuar ose

kur biznesi kërkon që një person të marr vendimet. Kjo teknikë kërkon më shumë investim në teknologji, por pasi investimi bëhet, numri i njerëzve të përfshirë në integrimin e aplikimeve mund të reduktohet zakonisht.**[4]** Reduktimi i përfshirjes humane zakonisht redukton kostot dhe rrit besueshmërinë (reliability). Ky lloj integrimi nuk konsiderohet si mjaftueshëm efiçent, duke qenë se akoma kërkon ndërhyrjen e disa personave në mes të proçesit, psh. nevoja e ndërhyrjes humane për të transformuar të dhënat që kërkohen nga një sistem tjetër, dhe gjithashtu të sigurohemi që të dhënat po përdoren nga aplikimi tjetër përpara se sa të hidhen.

Integrimi automatik i aplikimeve

Integrimi automatik është një teknikë që redukton varësinë humane në proçesin e integrimit dhe lejon aplikimet të ndërveprojnë pa ndërhyrje. Ky lloj integrimi konsiston në aplikime që komunikojnë nëpërmjet një tërësie ndërfaqesh (interfaces) dhe përshtatësish (adapters). Psh. dy databaza mund të ndajnë së bashku (share) të dhëna, që transformohen dhe kalohen nga databazë i parë në database-in e dytë pa ndërveprim human. Megjithëse integrimi plotësisht automatik i aplikimeve largon varësinë nga njeriu, sisteme të tilla mund të jenë më të shtrenjtë për t'u implementur dhe mund të mos jenë praktike për shumë probleme biznesi. Në shumë raste, një organizatë akoma do të kërkojë njerëz për të marrë vendime biznesi dhe shumë shpesh gjithashtu është më efiçente të ketë një person që kontrollon një proçes teknik. Teknika plotësisht automatike për integrimin e aplikimeve është bërë teknika më e besueshme (reliable) për shkak të faktit që shkallëzueshmëria, efiçensa dhe konsistenca derivohen në proçesin e ndarjes së të dhënave (data sharing), komunikimin dhe shkëmbimin e mesazheve. Teknikat e mësipërme përbëjnë përqasjet bazë për integrimin e aplikimeve. Secila teknikë ka avanatzhet dhe disavantazhet e saj, bazuar në mjedisin, natyrën e aplikimeve që kanë nevojë të integrohen dhe gjithashtu siguria dhe politikat e tjera të biznesit. në do të vlerësojmë kriteret dhe kërkesat për secilën teknikë dhe do të bëjmë krahasimin e tyre bazuar në kriteret dhe kërkesat në kapitullin pasues. Le të vështrojmë në vijim arkitekturën e integrimit që të shikojmë se si mund të adoptohet në teknikat e mësipërme.

Kapitulli 3 Arkitektura e Integrimit të Aplikimeve

Integrimi i aplikimeve mund të mbështetet nga arkitektura të ndryshme. Një nga arkitekturat është zgjidhja 'point-to-point' ku secili aplikim lidhet direkt me çdo aplikim tjetër. Kjo zgjidhje mund të funksionojë për një numër të vogël aplikimesh, por kur shtrihen shumë aplikime, numri i lidhjeve rritet shumë shpejt. Për më tepër, kur një aplikim i ri prezantohet, duhet të lidhet me çdo njërin prej aplikimeve të tjera. Arkitektura tjetër është mbi konceptin e modelit të integrimit bazuar në 'middleware'. Ideja kryesore është të reduktohet numri i ndërfaqeve duke vendosur një 'message bus' në qendër dhe nv këtë mënyrë e bën më të lehtë suportin ndaj ndërfaqeve. Nëse një nga aplikimet ndryshon formatin, vetëm një lidhje duhet të ndryshohet: lidhja me 'message bus. Në të njëjtën mënyrë, nëse shtohet një aplikim, vetëm një lidhje e re kërkohet.
[14]Një mënyrë shumë e rëndësishme për të bërë integrimin e aplikimeve të shkallëzueshëm është duke rritur numrin e hapave automatik dhe duke reduktuar numrin e hapave njerëzorë (manual). Një gjë e tillë e ka bërë integrimin automatik të aplikimeve më të sukseshëm, të shkallëzueshëm dhe efiçent sesa integrimet manualë dhe ½ automatik. Kjo implikon krijimin e ndërfaqeve ndërmjet aplikimeve për të zëvëndësuar punën njerëzore. Rritja e nivelit të automatizimit zakonisht rrit sasinë e informacionit që shkëmbehet ndërmjet aplikimeve pa qënë nevoja të rritet numri i punonjësve për të realizuar qëllimet e biznesit. Çështjet e shkallëzimit, gjithsesi nuk mund të zgjidhen thjesht nëpërmjet automatizimit. Duhet të konsiderojmë dhe numrin e aplikimeve dhe se si ndodh integrimi ndërmjet tyre. Në këtë punim do të trajtohen alternativat e mëposhtme për integrimin automatik:

1. Arkitektura 'Point-to-Point'
2. Arkitektura 'Middleware'
3. Arkitektura 'Service Oriented'

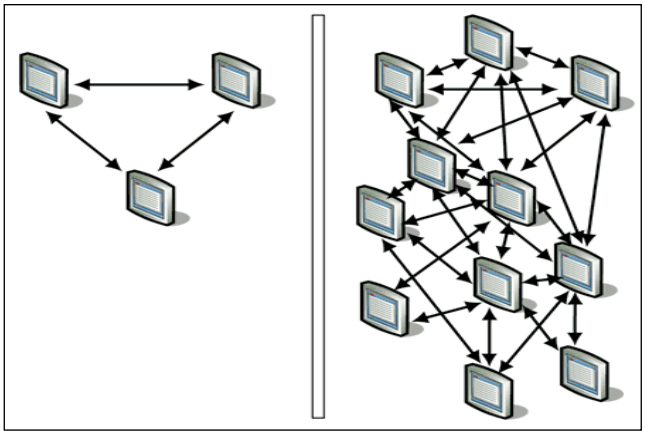
3.1 Arkitektura 'Point-to-Point'

Modeli 'point-to-point' përshkruan një strukturë të decentralizuar në të cilën çdo aplikim komunikon drejtëpërdrejtë me secilin prej aplikimeve të tjera. Me rritjen e numrit të aplikimeve dhe shërbimeve, rritet edhe numri i ndërfaqeve dhe lidhjeve që duhet të mirëmbahen në një mjedis point-to-point. Numri maksimal i lidhjeve që nevojiten përcaktohet si vijon:

∑x (x=1 to n-1)

Ku n është numri i aplikimeve që duhet të integrohen. Numri i ndërfaqeve që kërkohen mund të arrijë deri në 2-fishin e numrit të lidhjeve. Problemi në këtë metodë është mungesa e konsistencës. Sa më shumë që zhvillohen ose përditësohen aplikimet, aq më shumë ndërfaqe shtesë duhet të krijohen. Çdo ndryshim e bën mjedisin më të vështirë për t'u kuptuar, deri sa, arrin një pikë që struktura bëhet aq komplekse sa është e pamundur të menaxhohet efektivisht. Një

strukturë kaq komplekse mund të pengojë kompaninë që të ndryshojë (zhvendosë) qëllimet e saj të biznesit, për shkak të kostove të larta të ndryshimeve IT. Ky lloj integrimi është i përshtatshëm për kompani që duan të integrojnë pak aplikime me një numër të vogël shërbimesh.
Figura 13 tregon numrin e lidhjeve të nevojshme të krijohen në një mjedis point-to-point me 3 dhe 10 aplikime, në rastin kur duhet të sigurojmë që të gjithë aplikimet mund të komunikojnë me njëri-tjetrin.

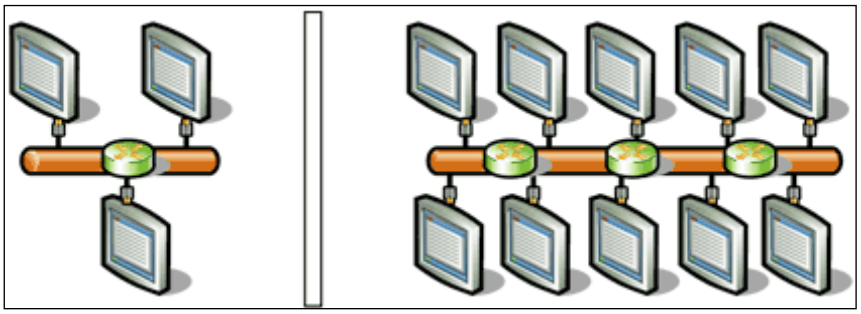


Figurë 13. Komunikimi 'point-to-point' ndërmjet aplikimeve

Në skenarin A të figurës 3, për 3 aplikime nevojiten 3 lidhje dhe 6 ndërfaqe, ndërsa për shembullin në skenarin B (10 aplikime) kërkohen 45 dhe dhe 90 ndërfaqe[4] **.**

3.2 Arkitektura 'Middleware'

Modeli 'middleware' ose 'integration hub' na siguron një strukturë më të centralizuar, në të cilën një 'integration hub' vendoset ndërmjet aplikimeve, dhe secili aplikim komunikon me 'hub' dhe jo drejtëpërdrejtë me aplikimet e tjera. Çdo aplikim ka nevojë vetëm për një ndërfaqe dhe një lidhje me 'integration hub'. Avantazhi kryesor i një mjedisi 'integration hub' është shkallëzueshmëria. Figura e mëposhtme tregon përdorimin e një 'integration hub' sërisht në mjedise me 3 dhe 10 aplikime.



Figurë 14. Aplikimet të lidhura nëpërmjet një 'integration hub'

Në figurën e mësipërme, kërkohet vetëm një lidhje me ‘integration hub’, me ndërfaqe të nevojshme tek aplikimi dhe tek ‘integration hub’. Megjithatë, shumë mjedise kërkojnë vetëm një ndërfaqe (ose asnjë nëse aplikimi përdor standartin e suportuar nga ‘hub’). Ky konfigurim na çon në një maksimum prej 3 lidhjesh dhe 6 ndërfaqesh për shembullin me 3 aplikime ose 10 lidhje dhe 20 nderfaqe në rastin e shembullit me 10 aplikime. Në më të shumtën e rasteve, numri i ndërfaqeve është ndjeshëm më i vogël. Modeli ‘integration hub’ është ndjeshëm më i shkallëzueshëm për mjedise integruese me shumë aplikime. Zakonisht kompanitë e mëdha kanë një numër shumë të madh aplikimesh. Është e pamundur të krijojmë ndërfaqe individuale për çdo pikë ndërveprimi. Zgjidhja është të krijojmë një mjedis integrues që lejon të gjithë aplikimet të komunikojnë në mënyrë logjike të paracaktuar. Një infrastrukturë e tillë ‘hub’ na lejon të modifikojmë ose përditësojmë elementët shumë më lehtë, dhe të bëjmë një veprim të tillë atëherë kur biznesi e kërkon dhe jo kur teknologjia paraprirëse të na e mundësojë. Gjithashtu e lejon kompaninë të ndryshojë më lehtësisht drejtimin e saj dhe të përdorë produktet dhe shërbimet e saj për të përmbushur kërkesat vazhdimisht në rritje. Mqs ndërfaqet ndodhen në ‘integration hub’ (dhe zakonisht janë ‘standard-based’), nuk është e nevojshme t’i rishkruajmë ato sa herë që prezantohen aplikime të reja. Megjithatë, implementimi i një ‘hub’ mund të jetë teknikisht një sfidë dhe madje shumë i kushtueshëm për mjedise më të thjeshta integruese. Akoma më tej, mund të duhet të sakrifikojmë kompleksitetin e disa të dhënave për të siguruar që çdo aplikim është konform standarteve të ‘integration hub’.**[4]**

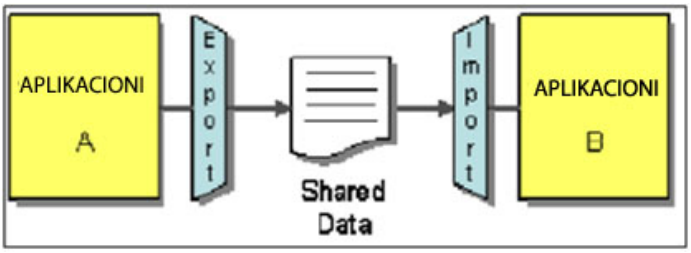
3.3 Stilet e integrimit

Ka shumë stile që sugjerohen dhe aplikohen për integrimin e aplikimeve dhe që variojnë nga ato në nivel të ulët deri tek nocioni i mesazheve asinkrone dhe orientimi drejt shërbimeve (web services). Këto trajtime mundësojnë integrimin duke ndarë (share) të dhënat ose logjikën e biznesit. Katër zgjidhje në nivel të ulët (low level), të cilat mund të duken të vjetra dhe të thjeshta kur krahasohen me teknologjitë e reja, janë identifikuar në këtë punim. Megjithëse të vjetra dhe të thjeshta, këto katër zgjidhje sic dhe do diskutohen në këtë seksion, janë akoma të zbatueshme dhe të dobishme. Ata gjithashtu përbëjnë bazën për skemat e tjera më të përpunuara/zhvilluara të arkitekturave integruese. Trajtimet që do diskutohen janë:

1. ‘File Transfers’
2. ‘Shared Databases’
3. ‘Remote Procedure Invocations’
4. ‘Messaging’
5. ’Web Services’

3.4.1 'File Transfer'

Synimi i kësaj arkitekture teknike të thjeshtë është të shkëmbehet informacion ndërmjet aplikimeve ose platformave të shumta nëpërmjet transferimit të skedarëve (files) në batch mode. Një aplikim e bën një gjë të tillë duke eksportuar informacionin e nevojshëm në një format standart. Pavarësisht nga mënyra, rezultati është një skedar me gjithë informacionin e nevojshëm, që më pas mund të transmetohet tek aplikime të tjera, të cilët më pas duhet të importojnë dhe filtrojnë informacionin nga skedari përpara se sa ta proçesojnë atë. Figura e mëposhtëme përshkruan skemën file transfer (export-import) ku aplikimi A shkëmben të dhëna me aplikimin B. Të dhënat transferohen ndërmjet sistemeve në mënyrë asinkrone duke përdorur batch file transfers dhe rrjedha e të dhënave është njëdrejtimëshe. Ndryshimet në njërin sistem shumë rrallë prekin sistemet e tjera, përveç se kur struktura, formati dhe semantika e të dhënave që po transferohen ndryshon. File transfer është një zgjidhje me kosto efektive për transferimin e batch files ndërmjet paltformave me variacion të gjerë.[4]



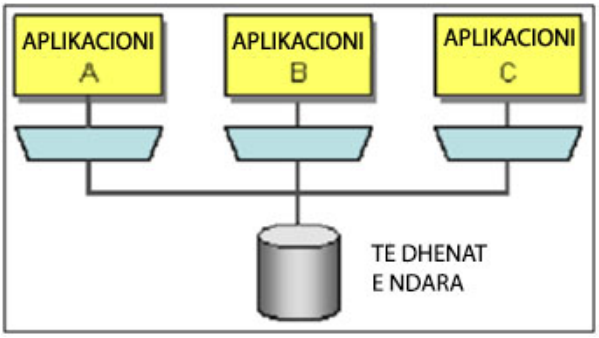
Figurë 15. Skema 'File Transfer'

Në këtë arkitekturë tipike, integruesit marrin përgjegjësinë e transformimit të skedarëve në formate të ndryshme, dhe garantojnë që skedarët gjenerohen në intevale të rregullta në përputhje me natyrën e biznesit. Një vendim i rëndësishëm me skedarët është çfarë formati të përdoret.[12] Shumë rrallë do të ndodhë që outputi i njërit aplikim të jetë ekzaktësisht çfarë nevojitet për aplikimin tjetër, për rrjedhojë kjo kërkon të bëhet pak proçesim i skedarëve gjatë rrugës. Jo të gjithë aplikimet që përdorin një skedar duhet ta lexojnë atë, një prej tyre mund të duhet të përdorë mjete proçesimi mbi të. Si rezultat, formatet standart të skedarëve po përdoren gjithnjë e më shumë me kalimin e kohës.

3.4.2 'Shared Database'

Skema file transfer e diskutuar më lartë ka tendencën të prezantojë një formë ngadalësie (latency) që është shpesh e pa pranueshme. Ndonjëherë, aplikimet duhet të punojnë mbi të njëjtin dataset gjatë kohës së tyre të operimit. Shared Database ofron zgjidhjen. Arkitektura "Shared Database" mundëson integrimin e aplikimeve të ndryshme duke bërë që aplikimet t'i ruajnë të dhënat e tyre në një databazë të vetëm shared. Nëse një tërësi aplikimesh të integruara mbështeten në

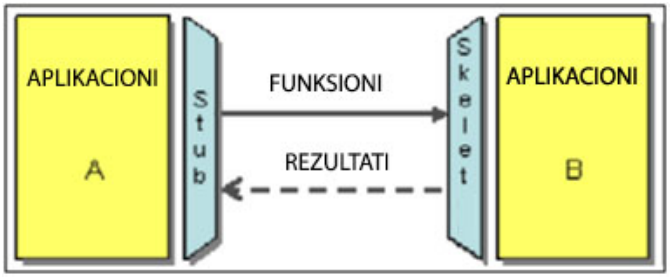
të njëjtën databazë, atëherë jemi mjaftueshëm të sigurtë që këto aplikime janë konsistente gjatë gjithë kohës. Nëse merren update-e të njëkohëshme tek një pjesë e vetme të dhënash nga burime të ndryshme, atëherë duhet të kemi sisteme për menaxhimin e transaksioneve që duhet të merret me këtë problem. Duke qenë se koha ndërmjet update-ve është shumë e vogël, gabimet mund të gjenden lehtë dhe të rregullohen. Database i "përbashkët" nuk duhet detyrimisht të jetë një databazë fizik; mund të jetë dhe një shared database virtual, në sensin që një mekanizëm komunikimi garanton që të gjithë instancat e ndryshme fizike të databazës mbahen të sinkronizuara. Menaxhimi i transaksioneve luan një rol të madh në këtë rast, por transaksionet janë mjaft të zakonshme për çdo DBMS. Skema ilustrohet si më poshtë:



Figurë 16. Skema 'Shared Database'

3.4.3 'Remote Procedure Call'

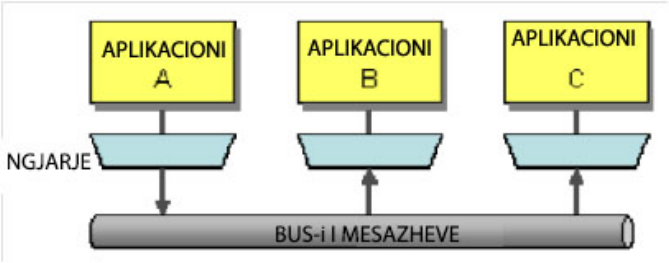
Skema Remote Procedure Invocation (RPI) e paraqitur në figurën 16, në ndryshim nga dy skemat e diskutuara më parë, që ndajnë vetëm informacionin, ndahet dhe në funksionalitete në më të shumtën e rasteve. C/C++ RPC dhe Java RMI janë që të dyja shembuj të Remote Procedure Invocation.[12] Remote Procedure Invocation aplikon parimin e enkapsulimit për të integruar aplikime të ndryshme. Nëse një aplikim ka nevojë për informacion që zotërohet nga një aplikim tjetër, atëherë e pyet drejtëpërdrejtë atë aplikim. Nëse një aplikim ka nevojë të modifikojë të dhënat e një tjetri, atëherë e bën një gjë të tillë duke kryer një thirrje (call) tek aplikimi tjetër. Secili aplikim mund të mirëmbajë integritetin e të dhënave që zotëron. Gjithashtu, secili aplikim mund të ndryshojë të dhënat e tij të brendshme pa prekur aplikimet e tjera. Parimi bazë i kësaj skeme është të zhvillohet secili aplikim si një objekt ose komponent 'large-scale' me të dhëna të enkapsuluara dhe rrjedhimisht të ofrohet një ndërfaqe për të lejuar aplikimet e tjera të ndërveprojnë me aplikimin në ekzekutim.



Figurë 17. Skema 'Remote Procedure Invocation'

3.4.4 'Messaging'

Messaging pretendon të jetë trajtimi teknik më fleksibël i integrimit krahasuar me skemat e mëparshme të transportit të diskutuara më lart. Kjo skemë akomodon ndarjen e informacionit si dhe ndarjen e funksioneve dhe shërbimeve. Gregor Hohpe dhe Bobby Woolf e përcaktuan këtë skemë si një përqasje (approach) teknikë e përdodur për të "transferuar paketat e të dhënave në mënyrë frekuente, me shpejtësi, me besueshmëri dhe në mënyrë asinkrone, duke përdorur formate të customizuar".[12] 'Asynchronous messaging' është krejtësisht një reagim pragmatik ndaj problemeve të sistemeve të shpërndara. Të dërgosh një mesazh nuk kërkon që të dy sistemet të jenë gati dhe në gjendje punë në të njëjtën kohë. Ideja bazë e messaging ilustrohet në figurën 17.



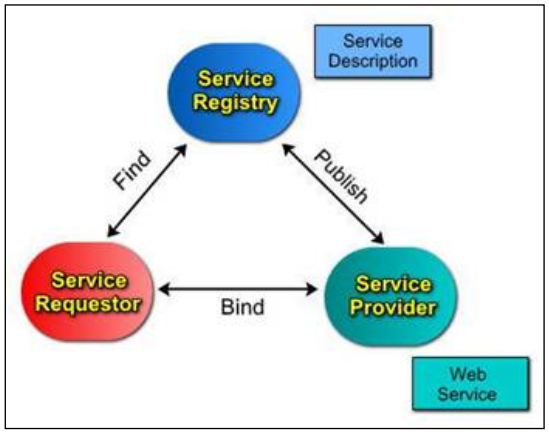
Figurë 18. Skema 'Message Bus'

Ndarja e informacionit është fare e thjeshtë me një skemë të mesazheve të vendosur ku ndryshimet transmetohen "live" tek aplikimi tjetër. Në këtë rast, "ngjarja" siç është modeluar në figurën më lartë do të jetë njoftimi i update.

3.4.5 'Web Services'

Një përcaktim për "Web service" nga W3C është si vijon: "Një web service është një sistem software i projektuar për të mbështetur ndërveprimin makinë-makinë në rrjet. Ka një ndërfaqe që përshkruhet me anë të një formati të proçesueshëm nga makina (specifikisht WSDL). Sistemet e tjera ndërveprojnë me web service në një mënyrë të parashkruar në përshkrimin e tij duke përdorur mesazhe SOAP, që zakonisht transportohen duke përdorur HTTP me një serializim XML". Çdo

funksionalitet ose shërbim mund të ekspozohet si një web service. Për çdo shërbim që do të ekspozohet si web service:
Funksionaliteti duhet të përshkruhet në gjuhën WSDL (Web Services Description Language)
Konsumuesit e shërbimit ndërveprojnë me web service duke përdorur mesazhe SOAP.
Në mënyrë opsionale, lista e web service mund të ruhet në UDDI registries. Ofruesi i shërbimit duhet të regjistrohet në UDDI registry. Një web service zakonisht identifikohet nga një URI (Unified Resource Identifier). Një web service ka përcaktimet WSDL. Për të komunikuar me web service duhet të përdorim mesazhet SOAP, që janë mesazhe "XML-based" që transportohen nëpërmjet protokolleve të internetit si HTTP, SMTP dhe FTP. Shërbimet web mund të përshkruhen më mirë me diagramën e mëposhtme:



Figurë 19. Koncepti i 'Web Service'

Në këtë rast, kërkuesi i shërbimit është klienti i web service, dhe ofruesi i shërbimit është hostuesi i web service. Një kërkues bën kërkesa SOAP tek ofruesi dhe ky i fundit përgjigjet. Një service registry mund të mendohet si një databazë web service-sh. Ka përcaktimet dhe URI-të për web service-t. Zhvilluesit e web service-ve mund t'i publikojnë shërbimet e tyre në një service registry nëse duan. Pas kësaj dikush mund të bëjë kërkim (query) dhe të zgjedhë shërbimin nga registry. Duke qenë se një registry ka të gjithë informacionin që nevojitet për të zhvilluar një web client, zhvilluesit mund të kërkojnë dhe të gjejnë informacionin që ju nevojitet nga një service registry duke përdorur UDDI (Universal Description, Discovery and Integration)
Mënyra më e përhapur dhe që përdoret më së shumti për të realizuar një SOA është duke përdorur web service. Ndërkohë që SOA na ofron parimet, shërbimet web na ofrojnë strategjinë për të realizuar një SOA.
Web Service zhvillohen duke përdorur një set standartesh XML si WSDL (Web Service Definition Language), SOAP (Simple Object Access Protocol) dhe UDDI (Universal Description Discovery and Integration).Web Services krijohen duke përdorur mjedise zhvillimi dhe shpërndahen në Application Server.

Kapitulli 4 – Ndikimi i rrjetit në Bazat e të Dhënave të Shpërndara

4.1 Rëndësia e rolit të rrjetave në arkitekturën SOA

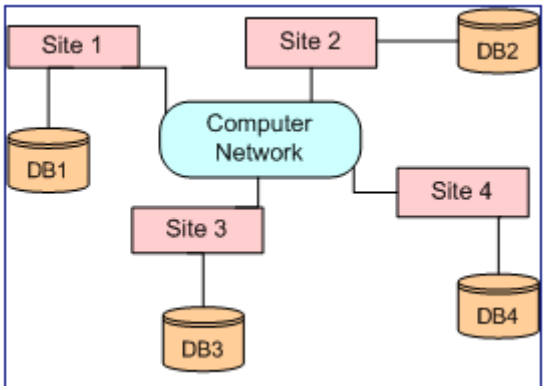
Grupet që merren me manaxhimin e infrastrukturës se IT, po vazhdojnë të mbajnë kërkesat e trashëgimisë dhe aplikacioneve të reja në një qendër të dhënash. [16]
Organizimi i IT (ITO) i shërben biznesit në mënyrë të tillë që në momente presioni të shpërndaje me shumë duke përdorur të njejtat burime apo dhe me pak – rregullat e reja mbi sigurine po kërkohen në qeverisjen e korporatave dhe mbrojtjen e informacionit, dhe problemet me perputhshmerine, reduktimi i kostos, manaxhimi i performancës se biznesit, dhe proçesi i funksionimit të nder-bizneseve janë në qendër të vemendjes se drejtuesve. Gjithnjë e me teper, biznesi po drejtohet drejt IT për të permirësuar efiçensën e tij. Brenda mureve të ITO, ritmi i aplikacioneve të reja është duke u përshpejttuar. Duke u nisur nga kërkesat gjithnjë në rritje, zhvilluesit janë duke përshtatur metodologji të reja.
Zhvillimi si psh RAD (rapid application development) ose teknika ekstreme programimi, dhe grupet e programimit po lëshojne një lum programesh të reja dhe shërbime që ekzekutohen në rrjetin e korporatës dhe po aq mirë perhapen edhe drejt klienteve dhe partnereve të biznesit.
Në mbështetje të këtyre aplikacioneve - edhe çfarë është trashëguar dhe të rejat - pyetja është e njëjtë si do të mundet Organizata e IT, të arrijë objektivat e fleksibilitetit, sigurimin e aksesit në këto aplikacione duke siguruar disponueshmeri të vazhdueshme, përshkallëzim dhe performancë, është ose pengesë në kosto apo në shumë raste e pamundur për të dizenjuar këto funksione brenda vete aplikacioneve.Si rrjedhim, një rol thelbesor luan rrjeti në njesimin e aplikacionit dhe në varësinë ndaj infrastrukturës.

Kjo është një kthesë shumë e mirë nga e kaluara, ku rrjetit i jepej vazhdimisht një konsiderate dytësore dhe shihej thjesht si lidhës. Pa rrjetin e tij, biznesi nuk ka aplikacione, dhe pa aplikacionet e tij, biznesi nuk mund të funksionoje. Për shumë persona, humbja e rrjetit çon direkt në humbje të ardhurash. Gjithsesi tradicionalisht, rrjeti ka qene jofleksibel dhe i pamundur për t'ju përgjigjur sfidave të aplikacioneve të kohës, ose me hapesira sigurie, mungese përshkallëzimi, problemeve në performancë përgjatë WAN, apo dhe mos -funksionim.
Kjo ndodh pasi rrjeti ka mungese inteligjence për tu përshtatur dhe për të siguruar shërbime efektive në kosto të cilat mund të përdoren për llogari të aplikacionit. Për networker-at për të shpërndare aplikacionet në mënyrë të suksesshme, nuk është thjeshte çështje shtimi të kapacitetit apo lidhjes. Qasje të reja në shpërndarjen e aplikacioneve – si psh arkitekturat e orientuara drejt shërbimeve, peer-to-peer, dhe grid computing – ndryshojnë në mënyrë radikale rrjedhën e trafikut ndërmjet pikave në rrjet.
Qasja reaguese tipike për të adresuar sfidat e aplikacioneve në network është aplikimi i 'point solutions', si psh ekuilibruesit e ngarkeses së serverave, paisjet e

përshpejttimit, paisjet ndërhyrese, paisjet detektuese/parandaluese, ose zgjidhjet e transformimit të permbajtjes.
Kjo rezulton në kosto shumë të larta të supportit të infrastrukturës, ku problemi domain (applikacioni) është transferuar në rrjet në formen e shumë paisjeve të vecanta të cilat performojne në mënyrë një funksionale nga shites të ndryshem – që është pikerisht lloji i gabimit dizenjues arkitekturor që shmangin kompanite me të sprovuara në treg.
Kërkohet një nivel shumë i lartë i automizimit, integrimit dhe dizenjimit arkitekturor, me mbështetje inteligjence bazuar në rrjet.

4.2 Bazat e të dhënave të shpërndara

Pavarësisht llojit të aplikacionit apo shërbimeve të ndryshme, është shumë e rëndësishme që informacioni të cilin ato përdorin të ketë një qendrueshmeri të lartë, të jete i ruajtur, i sigurt, si edhe të ketë mundësinë e zgjerimit sipas kërkesave dhe rritjes së përdoruesve. Tendenca për adresimin e këtyre kërkesave është e prirur drejt bazave të të dhënave të shpërndara.[17] Ndryshe nga sistemet paralele, në të cilat, proçesoret janë shumë të lidhur dhe i referohen vetëm një sistemi baze të dhënash, një sistem i shpërndare bazash të dhënash, konsiston në site të lidhur dobët të cilët nuk ndajne me njëri-tjetrin komponente fizike. Për me teper, sistemet e bazave të te dhënave që ekzekutohen në çdo site mund të kenë një shkalle të konsiderueshme pavarësie të ndersjellte.
Secili site mund të marre pjesë në ekzekutimin e transaksioneve që aksesojne të dhëna nga një apo disa faqe. Dallimi me i madh ndërmjet sistemeve të qendërzuara dhe të shpërndara të bazës së të dhënave është se, në të parën të dhënat vendosen në një vend të vetëm, ndersa në të dyten, të dhënat pozicionohen në vende të ndryshme. Kjo shpërndarje e të dhënave është shkaku i shumë vështirësive në proçesimin e transaksioneve dhe proçesimin e kërkesave



Figurë 20. Sistemet e bazave të dhënave të shpërndara

4.2.1 Bazat e të dhënave Homogjene dhe Heterogjene

Ne sistemet e bazave të dhënave të shpërndara Homogjene, të gjitha sitet kanë të njejtin sistem software të manaxhimit të bazës së të dhënave, janë të vetedijshme

për njëra-tjetrën, dhe bien dakord të bashkepunojne në proçesimin e kërkesave të përdoruesve.
Ne një sistem të tille, sitet lokale dorëzojnë një pjesë të autonomisë së tyre në kuptimin e të drejtave të tyre për të ndryshuar skemat apo manaxhimin e bazës së të dhënave.

Ne të kundërt, në bazat e të dhënave të shpërndara Heterogjene, site të ndryshem mund të përdorin skema të ndryshme, dhe sisteme të ndryshme software për manaxhimin e të dhënave.
Sitet mund të mos kenë informacion për njëri-tjetrin, dhe mund të sigurojnë vetëm bashkëpunim të kufizuar në proçesimin e transaksioneve. Diferencat në skema janë shpesh here një problem shumë i madh për proçesimin e kërkesave (queries), ndërkohë që divergjenca në software bëhet pengesë për proçesimin e transaksioneve që aksesojne shumë site në të njëjtën kohe.

4.3 Vonesat në rrjet

Çdo kompjuter mund të ekzekutojë një numër arbitrar shërbimesh, dhe çdo shërbim është ndërtuar në një mënyrë të tillë që siguron se ai mund të shkëmbejë informacion me çdo shërbim tjeter në rrjet pa ndërhyrjen e njeriut dhe pa patur nevojën e kryerjes së ndryshimeve në vetë programin kryesor.
Arkitektura SOA ndan funksionet në komponente të mirëpercaktuara, të cilat zhvilluesit i bëjnë të aksesueshme si shërbime në rrjet. Kjo bën të mundur të ekzekutohet SOA në një shumëllojshmeri platformash të shpërndara, të cilat mund të aksesohen nëpërmjet rrjeteve të shumëllojsheme.
Ndarja e të dhënave ndërmjet aplikacioneve të ndryshme është thelbi i aplikacioneve te biznesit.
Është e natyrshme që impakti i infrastrukturës së rrjetit egziston në lidhje me komunikimin e shërbimeve të ndryshme në baza të dhënash të shpërndara.
Duke u nisur nga rekomandimet dhe proçedurat e vazhdueshmerise së biznesit, rekomandimet për aplikacione kritike janë për të vendosur bazat e të dhënave larg njëra tjetrës, ku në disa raste mund të jenë edhe në kontinente të ndryshme. Në këto raste problemet të cilat lidhen me vonesat në rrjet janë të pranishme

Ne mund të masim vonesat e një lidhjeje rrjeti duke përdorur paisje network-u, por vetëm kjo nuk na jep të dhëna reale rreth efektit që shkakton kjo vonëse tek përdoruesi.
Vonesat në rrjet dhe performanca në rrjet ndërmjet shtresës së aplikacionit dhe shtresës së bazës së të dhënave, mund të jenë faktore sinjifikante në kohën e përgjigjes. Ka një efekt i cili mund të matet dhe këto matje mund të përdoren për të ushqyer me informacion për përmirësim në infrastrukture në mënyrë që të reduktohen vonesat.

4.3.1 Vlerësimi i Algoritmit të Rrugëzimit

Vërtet shtimi i lidhjeve ose rritja e kanalit të komunikimit midis burimit dhe destinacionit redukton vonesat, por kjo shoqërohet me një kosto të lartë. Egzistojne metoda alternative të cilat vlerësojnë dhe krahasojne algoritmat të cilat përdoren nga protokollet e rrugëzimit.Algoritmi i rrugëzimit është elementi kryesor i cilin ndikon në perzgjedhjen e rruges midis dy nyjeve që komunikojne.Rruget jo të zgjedhur mirë mund të ndikojne në përdorimin jo efiçent të kanalit të komunikimit dhe kjo ndikon në vonesat totale[15] . Pas një krahasimi të bërë midis dy algoritmave të ndryshme rrugëzimi të përdorura në një rrjet LAN nisur nga transferimi i bazës së të dhënave në SQL Server 2008 me kapacitet 15 GB, rezulton se rrugëzimi QoS(rrugëzimi i cilesise së shërbimit) kundrejt algoritmit SPF(rruga me e shkurter e para) në total ka rezultate me të mira për eleminimin e bllokimeve (starvation), të cilat janë të lidhura drejperdrejt me performancën totale të rrjetit.Algoritmi QoS në kete rast ndihmon balancuesin e puneve në rrjet dhe optimizon përdorshmerine e rrjetit.

Në një mjedis të shpërndarë bazash të dhënash do të krijohet një shtresë ndërmjetëse si në paragrafin 7.3, e cila do të komunikojë me të gjithë databazat e shpërndara në rrjet duke ruajtur integritetin dhe performancën, pavarësisht nga specifikat apo numri i aplikacioneve heterogjene që ndodhen në këtë mjedis të shpërndarë.

Bashkësia këtyre të dhënave mund te jetë e shpërndarë fizikisht në vendndodhje të ndryshme të shpërndara ne rrjeta lokale, intranet-te apo me anë të internetit, midis të cilave shërbimet e bazave të dhënave komunikojnë me njëra-tjetrën.
Aksesimi i të dhënave lidhet direkt me komunikimin në rrjet, referuar transferimit te mesazheve me anë të shërbimeve broker mbi databazën SQL Server 2008, në një rrjet jo LAN,të implementar në sistemin MPLS mbi teknikën IPV6 duke përdorur teknikën me Flow Label, e cila ndryshe nga router-at normale përdor router-at MPLS dhe shfaq një performancë me të mirë në sasine e throughput-it se në ate me routera IP [22]

Kapitulli 5 – SOA, Virtualizimi dhe Cloud

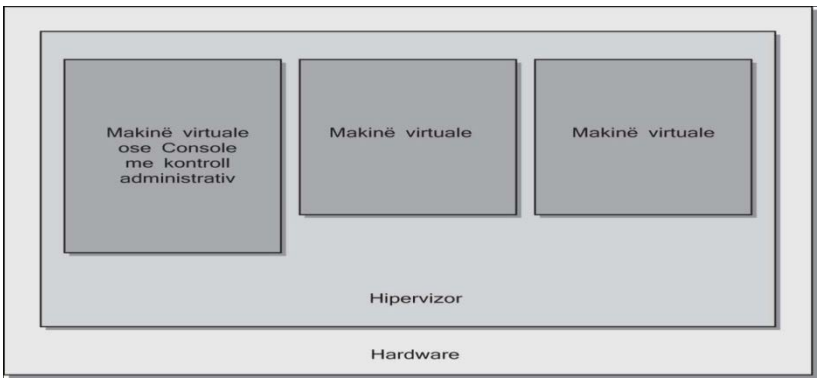
5.1 Virtualizimi

Virtualizimi është një ndarje logjike e kërkesës për shërbime nga burimet fizike që i sigurojnë këo shërbime. Virtualizimi siguron aftësinë për të ekzekutuar aplikacione, sisteme shfrytëzimi, ose sisteme shërbimesh në një mjedis të veçantë logjikë të pavarur nga mjedisi fizik i kompjuterit.Virtualizimi siguron një nivel logjik abstraksioni që lehtëson aplikacionet, sistemet e shërbimeve, madje dhe sistemin operativ nga të qenët të lidhura në një pjesë specifike të hardware-it. Fokusimi i virtualizimit në mjediset operative logjike në vend të atyre fizike i bën aplikacionet, shërbimet dhe instancat e një sistemi operativ të transferueshëm nëpër sisteme fizike të ndryshme kompjuterash.

Ka shumë tipa të ndryshëm të virtualizimit, të gjithë rrotullohen rreth idesë bazë të sigurimit të aksesit logjik në burimet fizike. Sot, virtualizimi është zakonisht i hasur në rrjete, sistemet e bazave të dhënave, dhe proçeset e serverave, në nivel të sistemit operativ dhe atë të makinës.

Virtualizimi mund të shihet si një pjesë e prirjes së përgjithshme të iniciativës IT që përfshin automatic computing, një skenar në të cilin mjedisi IT do të jetë i aftë të menaxhojë veten duke u bazuar në aktivitetin e perceptuar, dhe dobinë e kompjuterizimit, ku fuqia kompjuterike shihet si një dobi për të cilën klientët paguajnë për aq sa ju nevojitet. Qëllimi më i zakonshëm i virtualizimit është të qendërzojë detyrat administrative duke përmirësuar shkallëzueshmërinë dhe ngarkesën e punës.

Virtualizimi hardware i kompjuterit është virtualizimi i kompjuterit ose i sistemit operativ, i cili fsheh karakteristikat fizike të një platforme kompjuterike nga përdoruesi dhe në vend të kësaj tregon një platformë kompjuterike abstrakte. Në fillimet e tij software-i që kontrollonte virtualizimin u quajt “program kontrolli”, por në ditët e sotme ka marrë emrin “hypervisor” ose “monitor i makinës virtuale”. Hypervisor-i është administrator dhe menaxher i burimeve të përdorura nga makinat virtuale. Hypervisor-i si në figurën 21 vendoset në pjesën e sipërme të hardware dhe kjo quhet virtualizimi i plotë, ose mund të vendoset në pjesën e sipërme të sistemit operativ dhe kjo quhet virtualizimi OS.



Figurë 21. Diagrami logjik për makina virtuale të bazuara në hypervisor

Në varësi të tipit të hypervisor- it të përdorur dhe në përqasjen specifike të virtualizimit që ndiqet, kodi i burimit të sistemit operativ që vepron në një makinë virtuale mund te këtë nevojë për t'u modifikuar për të komunikuar me hypervisor.

5.1.1 Virtualizimi i shërbimeve

[18] Virtualizimi i shërbimeve është një nga idetë më të fundit i cili mund të sigurojë një kohë më të shpejtë të tregut për software, me cilësi më të lartë dhe risk më të vogël.
Virtualizimi i shërbimeve simulon sjelljen e komponentëve software, për të hequr kufizimet e varësisë në ekipet e zhvillimit dhe testimit, kështu që ata mund të ofrojnë software më të shpejtë, me risk dhe kosto më të ulët. Kufizimet e varësisë së sistemit mund të limitojnë seriozisht përpjekjet e zhvillimit dhe të testimit, veçanërisht në mjediset e ndërvarura kompleks. Komponentët downstream mund të jenë të padisponueshëm, me një performancë të dobët, ose të papërdorshëm. Përveç kësaj, koha dhe kostot për të ngritur skenarët për të dhënat test dhe laboratorët e testit fizikë së bashku me grindjet e shpeshta për burimet shared, mund të limitojnë në mënyrë të konsiderueshme qëllimin e testimit dhe kompromentojë cilësinë e aplikacionit. Virtualizimi i shërbimeve automatikisht kap dhe krijon modelet virtuale reale të sistemeve të pavarura, kështu që sistemet e vërtetë nuk nevojiten më për zhvillim dhe testim.
Termi "Virtualizim" është gjithashtu gjerësisht i përdorur në kontekste të tjera, siç janë virtualizimi i ruajtjes dhe virtualizimi i memories.

5.1.2 Historiku e virtualizimit të shërbimeve

Ekspozimi më i hershëm në virtualizim për shumicën e praktikuesve IT është puna me sistemet e virtualizimit të hardware siç janë Citrix ose VMware. Premisa e virtualizimit të hardware është e thjeshtë: hardware pak i përdorur është i zakonshëm për infrastrukturën IT për shumicën e organizatave. Duke bashkuar së bashku burimet hardware dhe kapacitetet e papërdorura, dhe duke alokuar këtë infrastrukturë të bashkuar për zhvilluesit e aplikacionit dhe testuesit, ndërmarrjet mund të eleminojnë kostot nga infrastuktura IT. Kjo është një ide e madhe e cila ka depërtuar gjerësisht në të gjithë ndërmarrjet në mbarë botën. Por a kap ajo të gjitha vlerat që priten për ndërtimin e teknikave të aplikacionit modem, siç janë SOA, BPM dhe mjediset e bazuara në Cloud(Cloud-based)?

Në ndëmarrjet e sotme të mëdha, vetë SOA nuk është më një dukuri e veçantë, në fakt, orientimi i shërbimit është baza për të pasur një proçes IT agile. Pyetja nuk është me të vërtetë se kush e bën SOA, por kush ofron shërbime cilësore më shpejtë dhe më lirë: ose, implementimi i modemit më të besueshëm, me kohën më të shpejtë në treg.

Zhvillimi i orientimit të shërbimit dhe testimi kanë evoluar shumë, tani kemi metodologji zhvillimi të reja dhe të sofistikuara, metodologji testimi dhe mjete që adresojnë nevojat e këtyre aplikacioneve të shpërndara, heterogjene, multi-tier. Ndërsa SOA flet për sisteme autonome që janë integruar dhe ka ende të lirë për t˙u

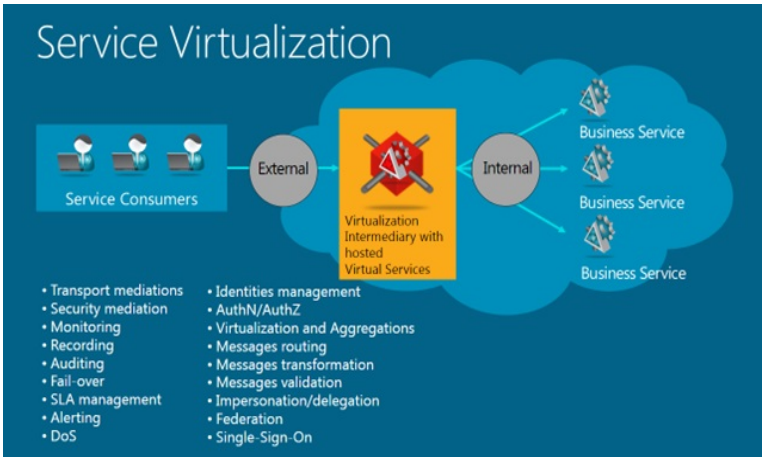
zhvilluar në mënyrë të pavarur, ajo nuk arrin afatet kohore të zhvillimit të ekipeve që duhet të ndajnë aksesin në sistemet shared. Projektet e mëdha të implementimit dhe integrimit janë të mbushur me sekuenca të detyrave që pëfshijnë shumë kufizime të sistemit shared që janë rezistente ndaj virtualizimit të hardware.

[23]Teknikat e zhvillimit service-based ofrojnë një ripërdorim më të madh të komponentëve të software për të përmbushur nevojat e biznesit. Sidoqoftë, kur përdoret vetëm, SOA nuk arrin të adresojë çështjen e kostove mjedisore të shpenzuara në testimin e këtyre sistemeve të integruara, për shkak të faktorëve siç janë koha e shpenzuar për shkak të setup-it kompleksë të të dhënave, koha e humbur për shkak të varësisë së padisponueshme, dhe kosto më të larta.
Virtualizimi i shërbimeve është aftësia për të imituar sjelljen dinamike dhe karakteristikat e performancës së një shërbimi në një mjedis virtual. Virtualizimi i shërbimeve mundëson replikimin e sjelljes për një aset IT i cili është shumë i vlefshëm kur sistemi target ka kufizuar vlefshmërinë apo koston e lartë të përdorimit. Për ekipet e testimit, zhvillimit dhe performancës, Shërbimi Virtual përgjigjet në sistemin në provë ashtu si komponentët live.

Virtualizimi i shërbimeve ka evoluar sot për të përfshirë jo vetëm shërbimet por gjithashtu sistemet e bazës së të dhënave, mainframe-t, dhe një gamë të gjerë të aseteve të tjera IT.

Virtualizimi i shërbimeve ka përparuar në një pikë ku ajo mund të përmirësojë në mënyrë të konsiderueshme parashikueshmërinë dhe shpejtësinë e një zinxhiri kritik në të dyja, testim dhe zhvillim.

Duke thyer kufizimet rreth sistemeve të mbi shfrytëzuar, virtualizimi i shërbimeve mund t'i mundësojë organizatës të marrë një shtresë të shërbimit që është me kosto më efektive, më fleksibël, dhe në gjendje për të ofruar aplikacionet më shpejtë në treg se konkurrenca.
[19] Virtualizimi i shërbimeve është koncepti që i lejon shërbimet fizike të ekspozohen përmes një aplikacioni ndërmjetës(ose broker) në mënyrë të ngjashme me mënyrën se si reverse proxy klasikë i përcjellin mesazhet të aplikacionet prapa tyre. Ndryshimi ndërmjet reverse proxy "primitivë" dhe ndërmjetësve të virtualizimit të shërbimeve është se këto të fundit janë të ngarkuar me inteligjencë dhe njohuri të asaj që shërbimet dhe API-të në të vërtetë janë duke ekspozuar. Kjo inteligjencë është ajo që i bën ndërmjetësit e virtualizimit një instrument të fuqishëm software dhe jo ndërhyrës në skenarët brokerage të shërbimit të jetës reale.
Konsideroni diagramën e mëposhtme ku një ndërmjetës shërbimi ekspozon potencialisht qindra shërbime biznesi sikur ata janë shërbime të vërteta, por me vlerë të shtuar të monitorimit jo ndërhyrës në kohë reale, autentifikimit të menaxhuar dhe kontrollit të aksesit, sigurisë dhe tejkalimin e protokolleve të komunikimit, menaxhimit 'Service Level Agreements' dhe shumë të tjerë në të njëjtën kohë.



Figurë 22. Virtualizimi i shërbimeve

5.2 Cloud computing

Cloud computing është një model që lejon përshtatjen e kërkesave të networkut për të aksesuar një bashkësi burimesh të konfigurueshme si (interneti, serverat, hapesirat, aplikacionet dhe shërbimet) të cilat mund të realizohen me një shpejtësi të madhe si edhe me një kërkesë minimale suporti.
Një sistem cloud computing ka një mënyrë punë të thjeshtë. Kompjuterat lokalë nuk kanë nevojë të bëjnë punën e rëndë të ekzekutimit të aplikimeve. Sistemi i kompjuterave në cloud merret me këtë gjë. E vetmja gjë që i nevojitet kompjuterit të përdoruesit është një ndërfaqe software e sistemit cloud computing, që mund të jetë një Web browser i thjeshtë. Pjesa tjetër mbetet në dorë të rrjetit cloud. E mira e cloud computing është se mund të fillosh të punosh me atë platformë në çdo moment. Ç'dokush me një lidhje interneti dhe me një kartë krediti mund të përdorë të njëjtat burime për të hostuar dhe për të ndërtuar apilakacionet web.

Ne në rolin si përdorues të internetit kemi përdorur disa forma të cloud computing. Nëse kemi një e-mail account si psh Hotmali, Yahoo! Mail ose Gmail, atëhere në kemi disa eksperienca me cloud computing. Në vend që të ekzekutojme një program e-mail në kompjuterin tonë, në logohemi remote në një Web mail account. Software-i dhe kujtesa fizike e account-it tonë nuk ndodhen në kompjuterin tonë personal, por në kompjuterin në cloud.

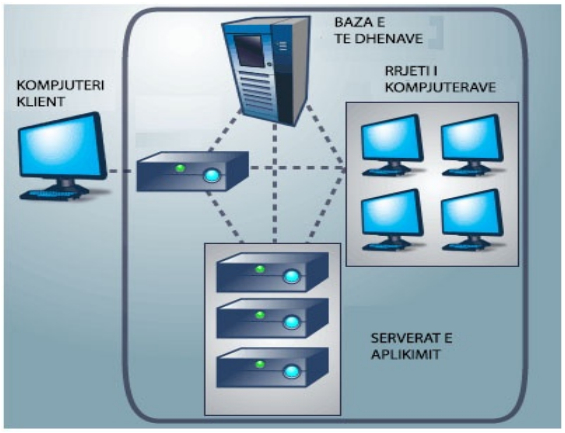
5.2.1 Arkitektura Cloud Computing

Kur flasim për një sistem cloud computing, është mirë ta ndajmë atë në dy seksione: **front end-i** dhe **back end-i**. Këto seksione lidhen me njëri-tjetrin me anë të një rrjeti, zakonisht Interneti. Front end-i është ana e kompjuterit të përdoruesit, ose ndryshe nga klienti. Back end-i është pjesa "cloud" e sistemit.

Front end-i perfshin një kompjuter klient dhe një aplikim të nevojshëm për të aksesuar sistemin cloud computing. Jo të gjithë sistemet cloud computing kanë të njëjtën

ndërfaqe. Shërbime si aplikime e-mail përdorin Web browsers, ndërsa sistemet e tjera kanë aplikacione të tyre për të siguruar lidhjen e klientëve në rrjet.

Në back end-in e sistemit ka kompjutera të ndryshëm, servera dhe sisteme të ruajtjes së të dhënave, të cilët sëbashku krijojnë "cloud"-in. Në teori një sistem cloud computing mund të përfshijë praktikisht çdo program kompjuteri që mund të imagjinojmë, që nga programe për proçesimin e të dhënave deri tek lojrat. Zakonisht çdo aplikacion ka serverin e tij të dedikuar.
Një server qëndror administron sistemim, monitoron trafikun dhe kërkesat e klientëve, duke u siguruar që çdo gjë të ekzekutohet normalisht. Kjo ndjek një bashkësi rregullash që quhen **protokolle** dhe përdor një lloj të veçantë software të quajtur **middleware**. Middleware bën të mundur që kompjuterat në rrjet të komunikojnë me njëri-tjetrin.



Figurë 23. Arkitektura Cloud Computing

Cloud computing nuk është vendi në të cilin aplikimi i akseson burimet në mënyrë të drejtëpërdrejtë, por ai i akseson ato me diçka të ngjashme me një shërbim (service). Pra, aplikimi në vend që të komunikojë me një hard drive për të dhënat dhe me një CPU për kalkulime, ai komunikon me disa shërbime (services) që ia sigurojnë këto burime. Shërbimi (service) bën lidhjen e çdo kërkese për burime, me burimet fizike, në mënyrë që t'ja ofrojë aplikimit. Shërbimi (the service) akseson një sasi të madhe burimesh fizike dhe mund t'i ndajë ato në mënyrë dinamike sipas nevojave. Sipas kësaj mënyre, nëse një aplikimi i nevojitet një numër i vogël burimesh, psh për kalkulime, atëherë shërbimi (service) do t'i caktojë një sasi të vogël, psh një CPU. Nëse aplikimi kërkon një sasi të madhe burimesh, atëherë shërbimi do t'i caktojë një sasi të madhe CPU, psh një grid me CPU. i gjithë ky proçes trajtohet nga shërbimi dhe jo nga aplikimi.

Arkitektura e cloud computing

Arkitektura cloud computing → Service Oriented Architecture
→ Middleware Architecture
→ Enterprise IT Architecture

Software Architecture

Arkitektura e cloud computing është një strukturë e një sistemi e cila bashkon burimet e cloud, shërbimet middleware, komponentet softwerike, geo-location duke u bazuar në vetite e tyre dhe në lidhjet që ekzistojnë midis këtyre komponentëve. Një pjesë e rëndësishme e arkitekturës është dhe dokumentimi i stemit të cloud. Dokumentimi lehtëson komunikimin midis palëve të interesuara për të komunikuar, apo edhe midis dokumentëve ku përshkruhen vendimet e hershme, të cilat mund të ripërdoren në një kohë të dytë.
Cloud computing është duke krijuar një epokë të re për IT duke ofruar një bashkësi shërbimesh, që duhet të kenë një kapacitet infinit. Ky është edhe si rezultat e evolimit të kompjuterit dhe tekonologjisë së komunikimit. Në këtë evolucion, fokusi zhvendoset nga koncepti i kompjuterit si diçka fizike në një data center si një bashkësi shërbimesh. Shumë organizata sot janë duke punuar me cloud në mënyrë që të ulin ngarkesën e të dhënave dhe zvogëlimin e kostos. Ndërkohë që zvogëlimi i kostos është një përfitim real, një vlerë është edhe rritja e shpejtësisë, fleksibilitetit dhe lehtësisë së shpërndarjes në mjedisin cloud. Por duhet theksuar se jo të gjithë janë duke e përqafuar cloud computing publik. Një mënyrë është edhe adoptimi në cloud computing privat. Një nga sfidat më të mëdha të cloud është pa diskutim baza e të dhënave dhe integrimi i saj në cloud. Gjithashtu privatësia e të dhënave dhe siguria e tyre janë arsye të mjaftueshme për t'i penguar sot bizneset që të përdorin cloud-in publik. Shumë probleme që sot paraqiten tek cloud publik janë të eleminueshme nga cloud privat, i cili konsiderohet si një pjesë e cloud publik. Kombinimi i fuqisë së kompjuterit dhe mundësia për ta aksesuar atë kudo që ndodhemi do të thotë më pak nevojë për të blerë dhe mirëmbajtur hardware.

Modelet shpërndarëse të cloud computing
Shërbimet që ofron sot cloud computing shpërndahen, transportohen nëpërmjet tre modeleve. Software as a Service (SaaS), Platform as e Service (PaaS), dhe Ifrastructure as a Service (IaaS). Lidhjet dhe pavarësitë ndërmjet këtyre tre modeleve janë shumë të rëndësishme. SaaS është shtresa bazë dhe është fondamentale për shërbimet që ofron cloud,PaaS është shtresa e mesme, kurse IaaS është shtresa e parë.

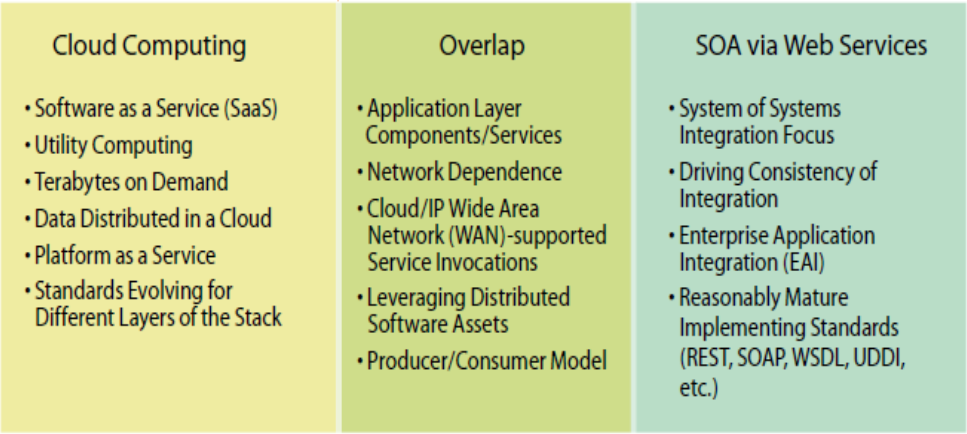
Software as a service (SaaS): Ky model i cloud computing që shpërndan aplikacionet tek konsumatori bazohet tek arkitektura e shumë qiramarrësve "multitenant". Konsumatorët janë në gjendje të përdorin shërbimet që "providerat" ofrojnë, të cilat ekzekutohen në infrastrukturën cloud, nëpërmjet një ndërfaqe si web browser. Përfitimi kryesor që kanë përdoruesit është se ata nuk e kanë të nevojshme të investojnë në servera ose në licensa. Konsumatori në këtë renditje nuk e ka të nevojshme të manaxhoje ose të kontrollojë infrastrukturën cloud, duke përfshirë këtu serverat, rrjetin, sistemet e operimit, hapësirën. Ky model është edhe më i përdorshmi, me të cilin përdoruesit janë më familjarë.

Platform as a Service (PaaS): Ky model shpërndarës i cloud computing zhvillohet në mjedisin e konsumatorit. Konsumatorëve u jepet mundësia të zhvillojnë në aplikacionet e infrastrukturës cloud duke përdorur gjuhët e programimit dhe mjetet e punës që suportohen nga ofruesi i shërbimeve. Këto aplikacione mund të ekzekutohen në infrastrukturën e ofruesit të shërbimit dhe me pas të shpërndahen tek konsumatori nëpërmjet internetit. Edhe ketu njësoj si tek SaaS, konsumatori nuk mund të manaxhojë dhe të kontrollojë serverat, sistemet e operimit, rrjetin, por në këtë rast ka mundësinë e kontrollit të shpërndarjes së burimeve dhe konfigurimin e mjedisit hostues.

Infrastrukturë as a Service (IaaS): Ky model i cloud computing shpërndan burime si serverat, lidhjet dhe mjetet e tjera të nevojshme. Konsumatori ka të drejtë të aksesojë shërbimet, hapësirat, rrjetin dhe burime të tjera të rëndësishme të cilat janë të nevojshme për të ekzekutuar aplikacionet. Këtu konsumatori nuk e ka mundësinë të menaxhojë apo të kontrollojë burimet e informacionit, por ka mundësinë e kontrollit të sistemeve të operimit, hapësirat, komponentet e rrjetit etj.

5.2.1 SOA dhe Cloud Computing

[20] Cloud computing dhe SOA kanë çështje të rëndësishme të mbivendosura dhe konsiderata të përbashkëta siç tregohet në Figurën 24. Mbivendosja më e rëndësishme ndodh në krye të raftit cloud computing, në hapësirën Cloud services, të cilat janë kopmonente aplikacioni të aksesueshëm përmes rrjetit dhe shërbime software si Web Services bashkëkohore.



Figurë 24. Koncepte të implementimit për Cloud Computing dhe SOA

Cloud computing dhe SOA ndajnë koncepte të orientuara drejt shërbimit. Shërbime të tipeve të ndryshme janë të disponueshme në një rrjet të përbashkët për t'u përdorur nga konsumatorët. Cloud computing fokusohet në kthimin e aspekteve të raftit IT computing në mallra(ose artikuj të përdorimit të gjerë) të cilat mund të blihen në mënyrë inkrementale nga cloud-based provider dhe mund të konsiderohen një lloj 'outsourcing' në shumë raste. Psh, ruajtja online në shkallë të gjerë mund të sigurohet

dhe të alokohet automatikisht në njësi terabyte nga cloud. Në mënyrë të ngjashme, një platformë për të operuar, aplikacionet e bazuara në web(web-based) mund të merren me qera nga qendrat e të dhënave redundant në cloud. Sidoqoftë cloud computing është një term më i gjerë se SOA dhe mbulon të gjithë raftin nga hardware përmes sistemeve software të presentation layer.
SOA edhe pse nuk kufizohet konceptualisht nga software shpesh implementohet në praktikë si komponentët ose shërbimet software sikurse ilustrohet nga standartet web service të përdoruara në shumë implementime.Këto komponente mund të jenë të lidhura bashkë dhe ekzekutohen në shumë platforma përmes rrjetit për të sigurur një funksion të biznesit.

- **Varësia nga rrjeti** – Të dyja, cloud computing dhe SOA mbështeten në një rrjet të fuqishëm për të lidhur konsumatorët dhe prodhuesit, dhe në këtë mënyrë të dyja kanë të njëjtën dobësi strukturore funksionale kur rrjeti nuk është performant ose kur është i padisponueshem.
- **Format e "outsourcing"** - Të dyja konceptet kërkojnë marrëdhënie kontraktuale dhe besueshmëri midis ofruesve të shërbimit dhe konsumatorëve të shërbimit. Ripërdorimi i një shërbimi SOA nga një grup i një sistemi tjetër është në efekt një "outsourcing" i të njëjtës aftësie të një organizate tjëter. Me cloud computing, "outsourcing" është më i hapur dhe ka shpesh shije plotësisht komerciale. Ruajtja, platformat dhe serverat merren me qera nga shpërndarës komercialë.
- **Standartet** - Të dyja, cloud computing dhe SOA ofrojnë një organizatë me mundësinë për të zgjedhur standartet e përbashkëta për aftësitë e aksesueshme të rrjetit. SOA ka një bashkësi standatesh për implementimin e shërbimeve software si Representational State Transfer (REST), SOAP, dhe Web Services Description Language (WSDL), midis shumë të tjerave. Cloud computing është jo aq e maturuar dhe shumë nga ndërfaqet e ofruara janë unike për një shitës të veçantë.

Ndërsa ka mbivendosje të rëndësishme mes cloud computing dhe SOA, ata kanë një theks të ndryshëm, si rezultat i fokusit të tyre mbi grupe me probleme të ndryshme. Implementimet SOA janë krejtësisht teknologji integrimi ndërmarrje për shkëmbim informacioni ndërmjet sistemeve të sistemeve.

SOA fokusohet në problemin për të bërë integrimin e sistemeve më efikas, dhe nqs integrimi sistemeve si një trend vazhdon të rritet, efiçenca në këtë detyrë do të bëhet gjithnjë e më e rëndësishme për ekipet e lidershipit Federal. Cloud computing e lejon tregun të ofrojë shumë funksione IT si artikuj të përdorimit të gjerë, duke ulur kështu kostot për konsumatorët.

5.2.2 Lidhja midis SOA dhe Cloud Computing

SOA dhe cloud computing janë aktivitete komplementare dhe të dyja do të luajnë role të rëndësishëm në planifikimin IT për ekipet e udhëheqjes së lartë për vitet që do të vijnë.

Cloud computing nuk e zëvendëson SOA, ose përdorimin e komponentëve software të shpërndarë, si një teknologji integruese. Nevoja për të suportuar integrim më të gjerë dhe më konsistent të sistemeve do të vazhdojë. Prirja nga ekipi i lidershipit për të marrë në konsideratë aftësitë IT si një mall do të vazhdojë të ulë presionin për buxhetin IT.
Cloud computing dhe SOA nuk janë sinonime, ndonëse ato ndajnë shumë karakteristika. Zgjidhja e njërit nuk e plotëson tjetrin. Psh, integrimi i vazhdueshëm i sistemit software si komponentët e shpërndarë ose shërbimet (SOA) në thelb nuk e virtualizon hardware, ose nuk e transferon shtresen e prezantimit tek një palë e tretë ofruese(cloud computing). Përmbushja e transferimit të sukseshëm të funksioneve IT të artikullit(cloud computing) nuk integron sistemet me porosi në biznes, ose të agregojë të dhëna në një shfaqje të vetme "mash-up"(SOA).

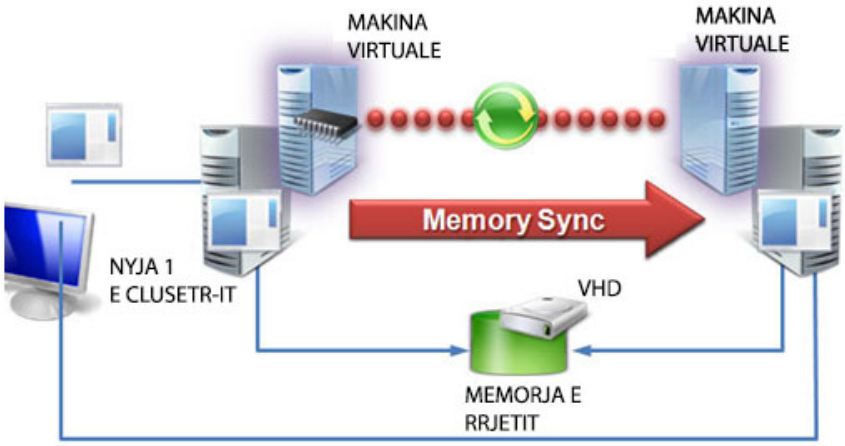
5.3 Cloud Computing dhe Virtualizimi

Çfarë e bën cloud computing vetvetja nuk është teknologjia hardware apo infrastuktura, çfarë e bën atë cloud computing është mënyra se si shërbimet dhe funksionet trajtohen dhe dërgohen. Në thelb ajo çfarë cloud computing do të thotë është që gjithçka bëhet nga një server i cili është i vendosur diku që nuk merremi më, ai është diku atje, "në re". Pas gjithë shërbimeve dhe aplikacioneve që janë emërtuar si "cloud" janë serverat, shumë e shumë servera ku të gjithë elementët proçesues janë CPU, RAM, GPU dhe madje edhe pajisjet e ruajtjes.
Në thelb virtualizimi ndryshon nga cloud computing sepse virtualizimi është software që manipulon hardware-in, ndërsa cloud computing i referohet një shërbimi i cili është rezultat i manipulimit.
[21] Cloud Computing është përtej teknologjisë së virtualizimit. Virtualizimi është i lidhur me kushtet në optimizimin e burimeve të infrastrukturës IT. Virtualizimi është një teknologji e përdorur në konceptin e Cloud Computing. Virtualizimi është duke përdorur të njëjtën infrastrukturë hardware për të ndërtuar disa servera për kërkesat. Virtualizimi përdor të njëjtën infrastrukturë hardware për të ndërtuar disa servera virtual për të cilët ka nevojë. Nqs kjo vendoset në një arkitekturë me shtresa, shtresa 1 do të jetë SAN (Storage Area Network), shtresa 2 do të jetë hardware servers(blade servers) për shpërndarjen e burimeve dhe shtresa top do të jetë host server. Software-t e virtualizimit si Citrix, VMware's vSphere, Xen, Microsoft Hyper V, Sun xVM do të ekzekutohen në serverat e shtresës top të cilët quhen host servers.

5.4 Teknika e Migrimit Live

Virtualizimi është bërë i zakonshëm në qendrat e të dhënave(data centers) moderne, shpesh të referuara si "computing clouds". Aftësia e migrimit live të makinës virtuale sjell përfitime si performancë të përmirësuar, përshtatshmëri dhe qëndrueshmëri ndaj dëmtimit, ndërkohë duke lejuar lëvizjen e workload me një downtime shërbimi të shkurtër.

Migrimi live është lëvizja e makinës virtuale nga një host fizikë tek një tjetër, ndërkohë që në mënyrë të vazhdueshme bëhet powered-up. Kur bartet siç duhet, ky proçes zhvillohet pa ndonjë efekt të dukshëm nga pikëpamja e përdoruesit fundorë.
Migrimi live e lejon një administrator të marrë një makinë virtuale offline për mirëmbajtje ose upgrade-tim pa nënshtruar përdoruesit e sistemit në downtime.
Një nga avantazhet më domethënëse të migrimit live është fakti se ai lehtëson mirëmbajtjen proactive.Migrimi live gjithashtu mund të përdoret për load balancing, në të cilin puna ndahet midis kompjuterave në mënyrë që të optimizohet përdorimi i burimeve të disponueshme CPU.



Figurë 25. Migrimi Live i makinave Virtuale

5.4.1 Migrimi live Hyper-V

Përdorimi i makinave virtuale është i mirë për të optimizuar burimet fizike. Pa virtualizim shumë servera Windows përdorin vetëm një pjesë të burimeve hardware që i janë caktuar. Kjo do të thotë që ka të ngjarë të blihet hardware i cili nuk përdoret kurrë. Hyper-V e ndryshon të gjithë këtë duke bërë të mundur që shumë makina virtuale të ndajnë një host të vetëm.
Por, ndërkohë që ndarja e burimeve do të thotë optimizim më të mirë, ajo gjithashtu paraqet potencialin për humbje më të madhe kur këta host-e nuk janë funksionalë. Kjo është një nga arsyet pse ekziston Migrimi Live. Proçesi i tij i migrimit të makinave virtuale e lejon një administrator të zhvendos VM-të përpara se të fiket një host për mirëmbajtje, patching, update, ose punë të tjera.Duke përdorur këtë proçes, pronësia e një makinë virtuale mund të transferohet në një numër të vogël sekondash dhe pa ndikuar në funksionimin e makinës virtuale.

5.4.1 Optimizimi i Migrimi live

[21] Për të bërë e mundur që kompani te ndryshme te cilat operojnë me infrastruktura te ndryshme IT, te operojne me efiçencë dhe t'u përshtaten ndryshimeve sipas kërkesave te biznesit perdorin aktualisht teknologjine e virtualizimit.

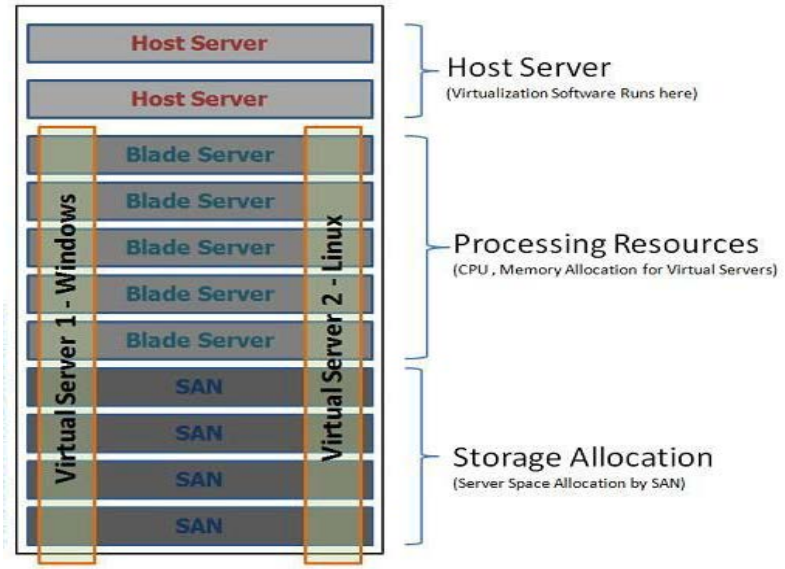
Gjithashtu disa prej tyre perdorin Arkitekturën e Orientuar drejt Shërbimeve (SOA) per te optimizuar kostot dhe investimet në burimet IT si dhe per te ofruar një fleksibilitet te lartë per t'ju përgjigjur ndryshimeve te mundshme në infrastrukturen egzistuese IT.
Cloud Computing është bashkesi teknikash te Virtualizimit,Arkitektures së Orientuar drejt Shërbimeve dhe perdorimit te Utility Computing.
Biznesi,perdoruesit qendrojne pas ketij koncepti.Ne vend që te kenë cdo Server,aplikim shërbim on-site, këto mund te merren nga ofruesi i një shërbimi Cloud kundrejt kushteve te ndryshme që lidhen me afatin,kohen dhe licensen e perdorimit duke optimizuar shumë faktore ku me kryesoret mund te permendim :

- Koston fillestare, mirëmbajtjen
- Qëndrueshmeri ndaj dëmtimit
- Performancën totale pasi fuqia perpunuese është në Cloud
- Vlefshmërine e Sistemeve te Informacionit

Nga ana tjeter që gjithe kjo skemë të funksionojë duhet që rritja e madhe e kërkesave nga klientet për burime Cloud të cilat vijnë gjithmonë e në rritje të ofrohet kundrejt një performancë dhe shërbimi te pranueshem.
Disa nga faktoret kryesore që ndikojne në performancën e Cloud mund të përmendim:

- Infrastruktura e rrjetit,Bandwidth
- Modeli i bazave te dhënave që mundeson te dhënat te jenë te shperndara
- Lloji i virtualizimit te perdorur,numri i serverave virtuale,teknika e migrimit live në Cloud
- Investimet te ndryshme

Me poshtë do të përqendrohemi në teknikën e migrimit live.
Kur diskutojmë për migrim, kuptojmë një rrjet të mirëfilltë serverash të lidhur me një sistem, baza të dhënash, ku janë ruajtur imazhet e sistemeve në rrjet. Sisteme të tilla të kombinuara së bashku në e si në figurën 26, formojnë atë që quhet datacenter e cila ofron "High Availability" ose vlefshmëri e lartë.



Figurë 26. Paraqitja skematike per një datacenter

Në të vërtetë gjatë migrimit live vihen re shumë bllokime të quajtura downtime gjatë kohës së migrimit të proçeseve nga një makinë e quajtur burim, në një tjetër të quajtur destinacion. Koha e bllokimit varet nga shumë faktorë si psh: Numri i proçeseve të cilët po ekzekutohen në çastin e migrimit, numri i faqeve të modifikuara dhe të paruajtura ende në disk, mënyra e dhënies së komandës së bllokimit, lloji i Hypervisor-it ku po punohet, rrjeti i komunikimit, etj.
Krijimi i një teknikë efiçente për rritjen e performancës të migrimit live midis makinave virtuale është perdorimi i një aplikacioni ndërmjetes midis shtresës hypervisor dhe makines virtuale i paraqitur si në tabelën 1:

Aplikacionet
Makina Virtuale
Aplikacioni në C
Hypervisor
Hardware

Tabelë 1. Aplikacioni i Integruar në Makinën Virtuale

Aplikacioni është i shkruar në gjuhen e programimit C dhe është i pavaruar nga ambjenti pasi është i aplikuar në hapesiren e perdoruesit.
Qëllimi kryesor i programit është te skanojne faqet “Dirty” dhe në baze te historikut te instruksioneve te perdoruara te reduktoje numrin e faqeve “Dirty” te cilat transferohen gjatëmigrimit live. Me reduktimin e faqeve te transferuara, permirësohet dhe koha totale gjatëmigrimit live midis makinave virtuale.
Ne tabelën 2 të paraqitur dhe në artikullin[21] jepen kohet në hipervizor te ndryshem, duke përdorur ose jo aplikacionin në gjuhen C.

Tipi Hivervisor	**Koha e Migrimit Live**	**Koha e Migrimit Live duke perdorur prog.C**
ESX-I	35 sec	29 sec
Hyper-V	45 sec	42 sec
XEN	38 sec	31 sec
OpenVZ	40 sec	34 sec

Tabelë 2. Krahasimi i kohës Live Migration për Hipervizorë të ndryshëm
Si konkluzion mund te themi që përdorimi i aplikacionit përmirëson kohën e migrimit live, kuptohet jo njësoj në te gjithë hipervisorët pasi kjo ka të bëjë me arkitekturën e secilit.

Kapitulli 6-Arkitekturat e Bazave të Dhënave të Orientuara nga Shërbimet (SODA)

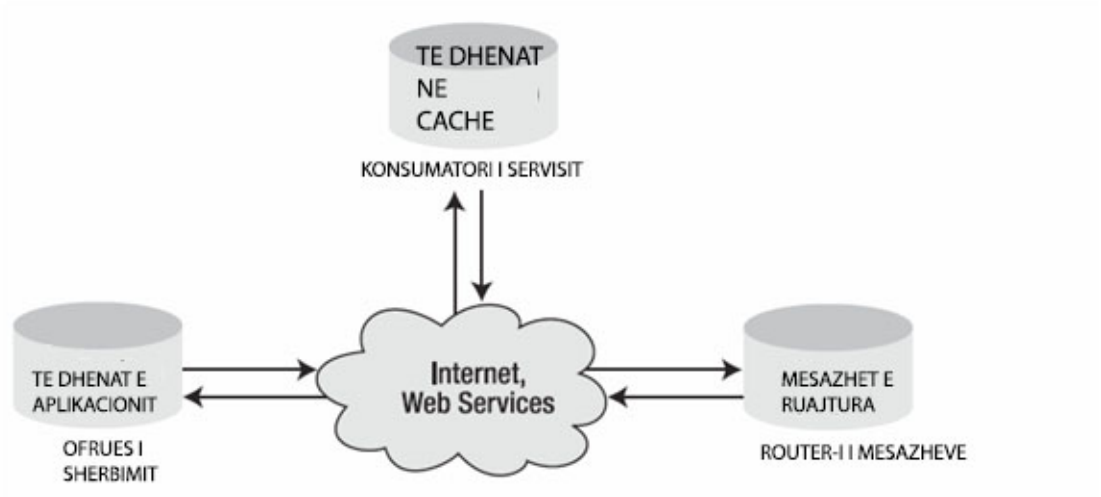
6.1 Prezantim me Arkitekturën SODA

Arkitekturat dominuese të aplikacioneve client/server dhe n-tier të dekades së fundit kanë ngritur një numër problemesh shumë serioze përshkallëzimi dhe disponueshmerie. Problemi me i madh ishte perpjekja për të ruajtur të dhënat në baza të dhënash masive të qendërzuara ku të gjitha komponentet klient kishin akses në to. Virtualisht i gjithe komunikimi me bazën e të dhënave ishte në formen e strukturave të SQL, të cilat prodhonin një bashkesi të dhënash si rezultat të një detyre specifike. Te tjera pobleme u ngriten gjatëperpjekjes për të inkorporuar sistemet e vjetra në aplikacione të reja.
Pas dekadash shpërndarjeje të një numri shumë të gjere sistemesh të ndërtuara në platforme dhe teknologji të ndryshme, bota ishte plote me sisteme që e kryenin punen e tyre në mënyrë perfekte por që nuk kishin asnjë shteg të qarte për të ndervepruar me aplikacione të tjera në një mjedis gjithmone e me tepër të orjentuar drejte rrjetit. Sigurimi i zhdervjellesise i kerkuar nga aplikacionet e sotme ka qene tepër e vështirë. Ndërveprimet Biznes-Biznes, i komplikojne gjerat edhe me teper, duke kerkuar standard, mënyra të besueshme të transmetimit elektronik të biznesit. në mënyrë të qarte, sistemet që permbushin nevojat e sotme të mjediseve të bisnëseve, kërkojne një arkitekture që përdor sistemet e trasheguara në mënyrë efiçente dhe sigurojnë një infrastrukture tregetimi të zhdervjellet.

SODA është një zhvendosje modeli i koncepteve të bazës së të dhënave dhe baza e të dhënave shihet përtej ruajtjes së të dhënave. Imagjinoni aplikacione që komunikojne me njëra-tjetrën nëpërmjet ndërfaqeve me logjiken e biznesit të inkorporuar dhe duke permbushur operacionet e deshiruara. Disa operacione mund të kërkojne marrjen e të dhënave nga sisteme të treta bazash të dhënash dhe të tjera mund të kërkojne perditesimin e të dhënave në një aplikacion tjeter. Gjithë kjo është e mundur me shkembimin e mesazheve në formatin XML ndërmjet sistemeve dhe shërbimeve. Sistemet tradicionale të bazës së të dhënave nuk kanë siguruar kete elasticitet, duke lene që e gjithe puna të bëhet nga zhvilluesi i aplikacionit. Arkitektura e bazës së të dhënave orjentuar drejte shërbimeve ka aktivizuar suportin për arkitekturën e aplikacionit të inkorporuar direkt në sistemet e bazës së të dhënave. Ky koncept kërkon ndryshime thelbesore brenda arkitektures tradicionale të bazës së të dhënave.

Kjo kërkon ekspozimin e të dhënave duke përdorur web service-et, duke lejuar shtresen e mesme dhe applikacionet e ngjarjeve të klientit për të komunikuar me bazën e të dhënave në një nivel të percaktuar shërbimi. këto shërbime duhet të nderlidhen nëpërmjet sistemeve komplekse të aplikacioneve të inkorporuara në sistemet e bazave të te dhënave.

Baza e të dhënave duhet të beje marreveshje me memorjen XML që është në SQL server. Logjika e biznesit në vend të shkruhet në procedura dhe paketa duhet të shkruhet si shërbime baze të dhanash. Kur të gjitha këto entitete integrohen, krijon të ashtuquajturen Service-Oriented Database Architecture (SODA).
Figura 26 tregon disa pika fundore që mund të krijojne një aplikacion jo fort të lidhur bazuar në SOA. një konsumator shërbimi, i cili mund të jete një aplikacion klient, një aplikacion server si psh një web server, apo çdo lloj aplikacioni tjeter, dërgon një mesazh në një ofrues shërbimesh. në sistemet komplekse, një mezash router-i mund të marre fillimisht mesazhin dhe aplikon disa veprime logjike për të drejtuar kërkesen tek ofruesi i duhur i shërbimeve.
Ofruesi i shërbimeve do të marre me pas mesazhin, ndoshta do ta cpaketoje dhe formatoje ate, do të kryeje çdo lloj punë të nevojshme për mesazhin, dhe me pas do të dërgoje një mesazh përgjigje tek konsumuesi i shërbimit.



Figurë 27. Aplikacioni i bazuar në shërbimet SOA

Detaji i rëndësishem në Figurën 27 është që, secila nyje në transaksion merr, ruan dhe transmeton të dhëna në forma të ndryshme. Ndonjehere të dhënat janë të përkohshme, herë të tjera secila nyje mund të vazhdojë të dhënat ose drejt cache-se ose në bazën e vetë lokale të të dhënave. Në driten e këtyre mënyrave të reja të manaxhimit të te dhënave brenda aplikacionit, baza e të dhënave në qendër të aplikacioneve SOA perballet me një set tjeter sfidash ndryshe nga ato që janë perballur me aplikacionet monolitike, n-shtresore. Integriteti i të dhënave është me i rëndësishem se kurrë më parë, por tashmë ka kërkesa shtese:

- Baza e të dhënave duhet të operoje në një mjedis ku kërkesat vijnë nga XML-based messages sesa nga lidhjet e dedikuara nga aplikacionet klientë.
- Depozitat e të dhënave të ruajtuara duhet të dine kur të bëjnë rifreskimin e të dhënave në mënyrë me dinamike sesa thjesht kryerja e një rifreskimi bazuar në një bashkësi skedulesh.

• Baza e të dhënave duhet të marrë pjesë në dialoget që ndodhin në një set sekuencash.
• Logjikat komplekse duhet të hostohen brenda ose afër bazës së të dhënave.

XML perben një format mesazhi shumë të mirë për sistemet e shpërndara gjeresisht. Pothuaj çdo system mund ta analizoje lehtesisht gramatikisht, dhe ka një gjuhe skeme – modelimi për të percaktuar strukturen ekzakte të te dhënave. Sistemet e shkëmbimit të mesazheve mund të bashkangjitin informacion në një mesazh XML në mënyrë të tillë që të dhënat të akumulohen në mesazh ndërkohë që ky i fundit rrjedh nga sistemi. Sistemet mund të analizojne gramatikisht dhe të proçesojnë çfarë kuptojnë dhe injorojnë pjesën tjetër. XML është një format shumë i përshtatshem për të suportuar sistemet e shpërndara.

6.1.1 Përdorimi i SODA

Microsoft njohu këto trende arkitekturore dhe ndërtoi tipe në SQL për të pjekur sfidat e reja ndërkohë që vazhdonte suportin e shumë aplikacioneve jo-SOA. Shumë prej tipeve të reja në SQL Server 2008 janë pjesë e një dizenjimi arkitekturor të integruar që suporton përdorimin e bazës së të dhënave si një ofrues shërbimesh SOA. Ka një numër arsyesh imponuese për të implementuar pjesët e SOA direkt në makinen e bazës së të dhënave, duke perfshire vijueset:

• Përshkallëzimi lartë dhe poshtë: madje edhe në aplikacionet me të medha enterprise SOA, një shërbim i vetëm mund të ilustrohet në thuajse çdo shkalle; një shërbim i përdorur rralle mund të ketë me pak aktivitet se në një baze të dhënash të vogel tipike. Integrimi me SQL Server nënkupton se një program shërbim mund të perfitoje nga suporti i të gjitheve për përshkallëzimin duke nisur nga paisjet e nderthurura deri tek serverat thelbesore të bazës së të dhënave, pa rritur kompleksitetin administrativ. Kodi logjik i shërbimit mund të ekzekutohet në çdo shkalle, dhe çdo implementim mund të përshkallëzohet jashtë – në një shtrese të mesme të veçantë nga koha e zhvillimit. Me SQL Server, logjika e shërbimit mund të përshkallëzohet jashtë jashtë – në një shtrese të mesme të veçantë nga koha e zhvillimit dhe mund të ekzekutohet ose në nivelin e të dhënave ose mund të pozicionohet në nivelin e mesem. Nëse dizenjoni një aplikacion me kujdes, duke percaktuar përshkallëzimin, ky mund të jete me tepër një vendim pozicionimi sesa një vendim dizenjimi apo kohe-zhvillimi

• Përshkallëzimi i jashtëm: mund të përshkallëzoni jashtë kompjuterizimit të centralizuar të te dhënave në mënyra të ndryshme, pergjithesisht ose duke përshkallëzuar jashtë bazës së të dhënave ose duke shpërndare proçesimin bazuar në principet e SOA. Përshkallëzimi jashtë bazës së të dhënave rezulton në një cluster baze të dhënash që është relativisht shumë e lidhur, ndërkohë që lidhja e service-oriented solution është me e dobët.

Mesazhet janë të dhëna

Mesazhet kërkese dhe përgjigje janë të dhëna që mund të kenë shumë vlere për tu arkivuar në bazën e të dhënave. Mbajtja e mesazheve në gjeendje të disponueshme me kalimin e kohës siguron një historik që lejon auditimin dhe analizen e transaksioneve të biznesit. Për shkak se mesazhet ruhen në tabela dhe kanë të disponueshme kataloget e view-ve, mund të përdoren lehtesisht T-SQL për të pare statusin e çdo pjesë të te gjithe sistemit

Kërkesat për një Ofrues Shërbimesh SODA (SODA Service Provider)

Për SQL Server 2008 apo dhe çdo makine baze të dhënash që luan rolin a një program shërbimi stand-alone SOA, duhet të implementoje disa pjesë të tjera shtese për të manaxhuar të dhënat:

• **Suporti fundor**: ofruesi i SODA duhet të siguroje suport për komunikimin për të marrë dhe transmetuar mesazhe, në mënyrë tipike si një TCP socket, HTTP GET or fundoret HTTP PUT, SOAP ose ndonjë tip tjeter fundori.

• **Proçesimi i shërbimit të kërkeses**: shumica e mesaxheve në SOA formatohen duke përdorur XML, keshtuqe ofruesi i shërbimeve duhet të jete në gjendje të proçesoje dhe mundësisht të transformoje të dhënat e mbyllura në forma të tjera sipas nevojës së komponenteve që perbëjnë shërbimin.

Ai duhet gjithashtu të jete në gjendje të marre pjesë në dialogje dhe bisedime komplekse sic merren dhe dërgohen mesazhet e pavarura nga një komponente tek tjetra. Duhet gjithashtu të jete në gjendje të marre pjesë në dialogje komplekse si merren dhe dërgohen mesazhet e pavarura drejte komponenteve të tjera

• **Hostimi logjikes së shërbimit**: ofruesi duhet të jete në gjendje të performoje çfarëdolloj logjike komplekse që kërkohet të proçesoje mesazhin dhe të siguroje përgjigjen e nevojshme, si dhe mundësisht të koordinoje inputin e disa shërbimeve të tjera.

Kjo mund të kërkoje detyra të perbashketa të serverit të aplikimit, si psh burimet e pooling aktivizimi dhe përshkallëzimi jashtë proçesimit logjik

Karakteristikat e reja të SQL Server 2008 sigurojnë support për këto funksione, pervec se shumë infrastrukturave të tjera për të suportuar manaxhuesin e të dhënave. Për shembull, një ofrues shërbimesh duhet patjeter të marre pjesë në një sistem SOA dhe të jete në gjendje t'i identifikoje veten të tjereve, të siguroje qendruseshmeri, të marre pjesë në komunikim dhe në transaksione, dhe të performoje karakteristikat e tjera në niveli aplikacioni.

6.1.2 Të dhënat në SODA

Gjate eksplorimit të koncepteve pas SODA, bëhet shpejte e qarte që çdo komponente në të gjithe sistemin perfshin marrje, proçesimin dhe transmetimin e të dhënave përgjatë funksioneve primare. Edhe nëse përgjigja e ofruesit të shërbimeve ndaj një mesazhi të dërguar nga një konsumator i shërbimit, është thjeshte të rrotulloje paksa diçka për të ndezur apo fikur diçka pa ndervepruar me bazën e të dhënave, ofruesi i shërbimeve duhet të proçesoje të dhënat në mesazh në mënyrë që të përcaktojë punen që duhet të kryhet. Gjithsesi, aplikimet modern të sistemit merren gjeresisht me të dhënat, keshtuqe është e zakonshme për një komponente SODA të ketë akses ose në një baze të dhënash lokale ose të centralizuar, ose tek të dyja në mënyrë frekuente.

Eshte e rëndësishme të bëhet një dallim ndërmjet të dhanave brenda shërbimeve dhe të dhënave jashtë shërbimeve.

Te dhënat e jashtme

[26] Të dhënat jashtë një shërbimi dërgohen ndërmjet shërbimeve si mesazhe dhe duhet të percaktohen në një mënyrë të kuptueshme nga të dyja: konsumuesi i shërbimit dhe ofruesi i shërbimit. Nevoja për të interpretuar të dhënat në të pakten dy shërbime të ndryshme e bën të nevojshme ekzistencen dhe disponueshmerine e një skeme të perbashket. Skema duhet të ketë gjithashtu disa karakteristika.

Se pari, përcaktim i një skeme të pavarur është e rëndësishme. Kjo do të thotë se dërguesi dhe marresi i një mesazhi duhet të jenë në gjendje të përcaktojnë skemen e mesazhit pa patur nevojën të konsultohen me njeri-tjetrin.

Se dyti, skema e mesazhit duhet të jete e zgjerueshme. Kjo lejon ofruesin e shërbimeve t'i shtojë informacion mesazhit përtej çfarë është specifikuar në skemë.

Te dhënat reference

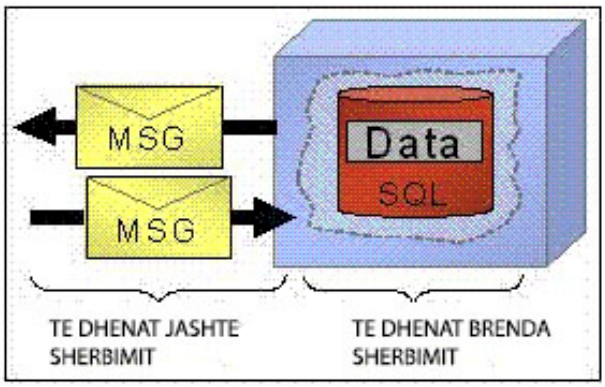
Te dhënat reference janë informacione të publikuara permes caqeve të shërbimit. Për çdo koleksion të të dhënave reference, një shërbim e krijon, e publikon, dhe në mënyrë periodike e dërgon tek shërbimet e tjera.

Te dhënat refence përdoren për të krijuar kërkesa shërbimi, si psh katalogu i produkteve. Të dhënat refencë duhet të jenë në një format, që është e përdorshme nga të dyja – nga ofruesi i shërbimeve dhe konsumatori dhe identifikohet në një mënyrë që nuk ndryshon me kalimin e kohës, sic është psh data e një katalogu.

Te dhënat reference mund edhe të importohen në një shërbim. Sapo lexohet në një shërbim, mund të proçesohet, riformatohet dhe indeksohet.

Te dhënat e brendshme

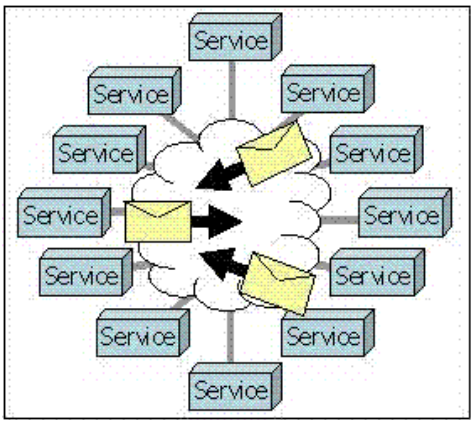
Ndryshe nga të dhënat jashtë shërbimeve, të dhënat e brendshme janë private për shërbimin siç tregohet edhe në Figurë n me poshtë. Te dhënat e brendshme janë gjithmone të enkapsuluara nga kodi i shërbimeve, prandaj e vetmja mënyrë për t'i aksesuar është nëpërmjet logjikes së biznesit të shërbimit. Të dhënat e brendshme perbehen nga tabela ku ruhen të dhënat transaksionale.



Figurë 28. Të dhënat brenda dhe jashtë shërbimeve

Mesazhet dhe Shërbimet

Një problem në SOA ka të beje me shërbimet e pavarura që perfshijne pjesë kodi dhe të dhëna, si dhe shërbimet nderlidhëse të mesazheve. Çdo shërbim është një koleksion unik i kodit dhe të dhënave dhe është i pavarur nga shërbimet e tjera. Gjithsesi, çdo shërbim është i nderlidhurme shërbimet e tjera nëpërmjet proçesit të mesazheve. Proçesi i mesazheve mban një rendesi shumë të madhe në SOA. Mesazhet dërgohen ndërmjet shërbimeve dhe rrjedhin ndërmjet tyre. Percaktimi i skemes për çdo mesazh dhe kontrate që përcakton rrjedhën e mesazheve specifikon sjelljen "black box" të shërbimit. Shërbimet janë shumë të lidhura me proçesin e mesazheve në mënyrë të tillë që çdo rruge hyrese apo dalese kryhet vetëm nëpërmjet mesazheve. Një shërbim partner është i ndergjegjshem vetëm për sekuencen e mesazheve që rrjedhin brenda dhe jashtë tij.



Figurë 29. Shërbimet dhe mesazhet të lidhura bashkë

Te gjitha shërbimet marrin mesazhe. këto mesazhe permbajnë veprime që i kërkojne shërbimit të performoje një funksion. Funksioni mund të jete një instruksion biznesi ose një urdher produkti. Funksioni mund të jete gjithashtu vetëm për të pranuar disa të dhëna reference të ardhura. Sapo një shërbim merr mesazhet, i ruan dhe i ekzekuton si të dhëna në një tabelë baze të dhënash. Kjo siguron që të dhënat ruhen dhe janë të përdorshme. në hapin tjeter, një tranaksion ekzekutohet dhe shenjon mesazhet e ardhura si të përdorura. Si pjesë e transaksionit, mesazhet dalese radhiten në baze të dhënash lokale për tu transmetuar me vone. Dhe në fund, i gjithe transaksioni ekzekutohet. Nëse transaksioni perfundon para se i gjithe veprimi të ekzekutohet, mesazhi hyres do të rishfaqet në radhe, dhe transaksioni do të riprovohet. Me probabilitet se një transaksion mund të abortoje, mesazhet dalese nuk dërgohen asnjehere deri sa transaksioni të proçesohet, për të siguruar që transmetimi i mesazheve është i pandashem me pjesën tjeter të transaksionit.

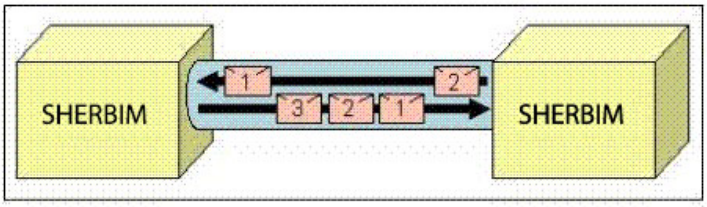
Nëse një mesazh është dërguar dhe me pas transaksioni aborton, mesazhi mund të proçesohet serisht edhe kur transaksioni deshton. Pandashmeria e transaksionit bie poshtë në kete situate, pasi transaksioni nuk është anulluar komplet.

Ndonjehere shumë mesazhe relevante dërgohen ndërmjet dy shërbimeve të ndryshme. Mesazhet relevante mund të rrjedhin midis dy shërbimeve gjatëjaveve ose muajve. Për shebull, një individ mund të rezervoje një bilete treni në 1 Qershor, 2011 dhe me pas të ndryshojë bileten për një date tjeter: në 5 Qershor, 2011. Individi mund ta konfirmojë më pas dhe të paguaje për rezervimin në 10 Qershor, 2011 dhe në fund të anulloje bileten për në 25 Qershor, 2011. Në këtë shembull, individi ka dërguar mesazhe disa dite për një numër të caktuar javesh. Kesaj i referohen si punë me kohë të gjatëekzekutimi.

Mesazhet në një punë të tillë janë relevante, ku mesazhi i dyte varet nga mesazhi i pare, dhe mesazhi i trete varet nga dy mesazhet paraardhes. një cookie ose diçka e ngjashme përdoret për të krijuar lidhjen ndërmjet mesazheve në një punë të tillë me kohë të gjatëekzekutimi.

Gjithashtu, fjala transaksion sugjeron një aktivitet me një fillim dhe një fund. Shumë punë me proçes të gjatëekzekutimi ka impakt në aplikacionet e tjera në

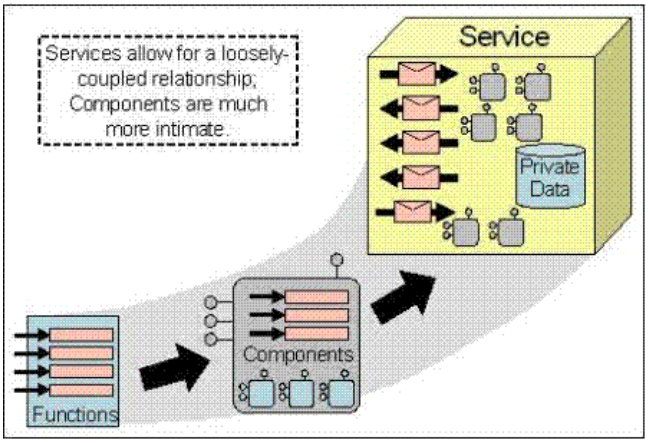
kuptimin që nuk ka një kufi të qarte kur një pjesë punë perfundon dhe një tjeter fillon.



Figurë 30. Mesazhe të shumëfishta të ngjashme

Shërbime dhe Komponentet

Ka ndodhur një evolucion natyral me kalimin e viteve duke perfshire funksionet, objektet dhe komponentet e shërbimet. në filim, kodi ishte i ndare në funksione që lejonin software-in të grupohej në pjesë me të vogla dhe me të organizuara. Me vone, komponentet dhe objektet u zhvilluan duke lejuar enkapsulimin e të dhënave. Tashme, shërbimet kanë arritur në një stad qendror në proçesin e evolimit. Shërbimet ofrojne një forme të trashe ndarjeje me me tepër pavarësi ndërmjet pjesëve të kodit sesa funksionet dhe komponentet. së pari, shërbimet nerveprojne gjithmone me njëra-tjetrën nëpërmjet mesazheve. Se dyti, janë jetegjate duke i lejuar ata t'i mbijetojne deshtimeve të sistemit apo restart-eve të tij. Se fundmi, shërbimet kanë ambient implementimi të padepertueshen ku vetëm ndërveprimet e proçesit të mesazheve janë të dukshme nga jashtë.



Figurë 31. Shërbimet kundrejt komponenteve

6.2 Mjete të Ms SQL Server për implementimin e koncepteve SODA

Një komponent kyc në çdo arktitekture të shpërndare, ndërtuar me metodologjine SOA është infrastruktura e bazës së të dhënave. Arkitektura e bazës së të dhënave orientuar drejt shërbimeve – SODA – është shumë here me e lehte për t'u

implementuar dhe shumë me e fuqishme me karakteristika të prezantuara në SQL Server 2005 dhe në leshimn e 2008. Teknologji si Service Broker, Native Web Services, Query Notifications, dhe SQL CLR ndërtuar brenda servisit SQL, siguron aftesine për të realizuar komplet arkitekturën SODA.

6.2.1 Suporti i XML

SQL Server 2005 ishte versioni i pare i SQL Server që suporton të dhëna ruajtese të formatit XML direkt në bazën e të dhënave. në versionin e meparshem të SQL Server, duhet të ruheshin të dhënat XML në tipe të dhënash si psh NVARCHAR. Tipi i të dhenes XML lejon të ruhen dokumente në formatin XML dhe fragmente në kolone. një fragment XML është një instancë XML që i mungon një element i vetëm të nivelit të lartë.mund të krijohen kolona, parametra, dhe variabla të tipit të te dhenes XML dhe të ruahen instancat e XML brenda tyre. Mundet gjithashtu të lidhen një koleksion skeme XML me një kolone, një parameter, ose një variabel të tipit të te dhenes XML. Skemat në koleksion përdoren për të validuar dhe për të percaktuar tipin e instancës XML të ruajtur në tipin e të dhenes XML. në kete rast, XML thuhet se ka tip të percaktuar, perndryshe nuk ka tip. Tipi i të dhenes XML siguron metoda për të thirrur dhe transformuar të dhënat XML direkt brenda SQL Server 2008. Metodat janë të besueshme, duke konsideruar se Service Broker dërgon mesazhe si të dhëna XML. në kete rast, mund të proçesohen dhe krijohen mesazhe për shërbime të tjera Service Broker direkt brenda SQL Server 2008 pa lejuar SQL Server 2008 të proçesoje mesazhe të tjera.

6.2.2 Native Web Services

Për vite me radhe, e vetmja mënyrë reale komunikimi ndërmjet klientit dhe serverit që ekzekutonte SQL Server ishte duke përdorur protokollin Tabulator Data Stream (TDS) të SQL Server. TDS ende është metoda e aksesimit të te dhënave me e shpejte dhe me efiçente, por për të komunikuar me serverin, klienti duhet të ketë të instaluar librarite perkatese. Ndonjehere, klientet e SQL Server duhet të përdorin Microsoft Data Access Components (MDAC). – komponentet e aksesimit të te dhënave të Microsoft. Paketa MDAC instalohet në kompjuterin e klientit që lidhet me SQL Server. Për SQL Server 2008 që të bëhet një ofrues i kompletuar i shërbimit SODA, duhet të suportoje protokollet standarte si dhe nyjet e nevojshme për pranimin dhe proçesimin e kërkesave të shërbimit për çdo lloj konsumatori të shërbimit.

Zgjerimi

Zgjerimi i SQLXML në SQL Server shtroi themelin e kesaj karakteristike duke perfshire një filtrim Internet Server API (ISAPI) për t'u përdorur me Internet

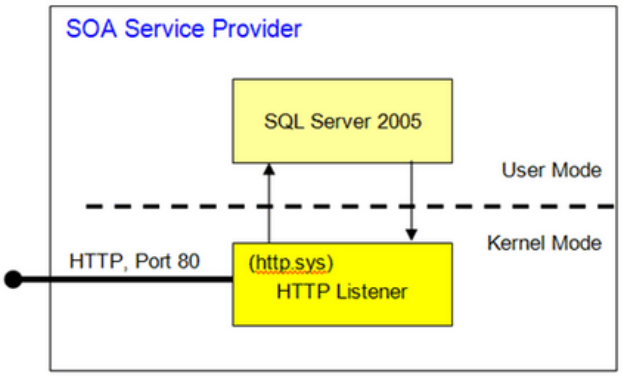
Information Services (IIS) për të lejuar komunikimin e HTTP web service me SQL Server. SQL Server 2008 perfshin një abstraksion nyjeje formale, e cila mund të përdoret për të suportuar tipe të ndryshme nyjesh, duke perfshire TDS, pasqyrimin e bazës së të dhënave, shërbimet e webit dhe Service Broker. Nyja e HTTP lejon instancën e SQL Server për të sherbyer si ofrues shërbimesh për çdo lloj aplikacioni në çdo lloj makine që ka support për web service-et nëpërmjet HTTP.

http.sys

Aksesimi i web servce-ve kërkon që SQL Server 2008 të instalohet në Windows Server 2003 për të perfituar nga driver-i kernel-mode i Windows Server - http.sys.

SQL server-i mund të regjistroje në kete driver të tipit kernel për të rezervuar porcione të namespace-eve të URL.

[27] Figura 32 tregon integrimin e ngushte dhe lidhjen direkte ndërmjet driver-it të tipit kernel dhe bazës së të dhënave SQL Server. Kur degjuesi i HTTP detekton një kërkese të HTTP në një porte të konfiguruar, ajo e drejton kërkesen direkt tek nyja që percaktohet brenda SQL Server. SQL Server me pas çfarëdolloj proçesimi të kërkohet dhe kthen një përgjigje mbrapsh tek thirresi.



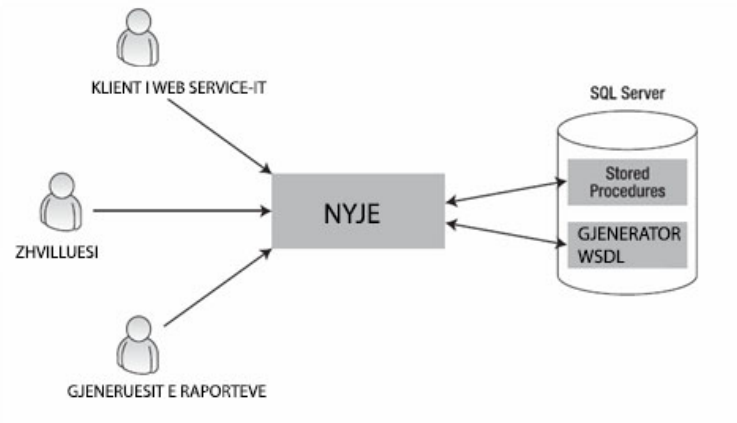
Figurë 32. Integrimi i SQL Server 2005 me driver-in http.sys

Për shkak se instalimi nuk kërkon IIS dhe për shkak se Kërkesat janë dërguar direkt nga driver-i i modelit kernel http.sys tek SQL Server, kërkesa të tilla janë efiçente dhe të thjeshta për tu administruar. http.sys siguron izolim të proçesit të kernel-based ndërmjet aplikimeve të ndryshme që mund të zoterojne porcione të ndryshme të namespace-ve të URL. Rrjedhimisht, nyje të ndryshme në të njëjtën instancë apo në instancë të ndryshme të SQL Server, nuk mund të interferojne me njëra tjetren.

Nyjet http

Për të përdorur web service-et lokale në SQL Server 2008, një nyje HTTP duhet të instalohet në server. Kjo nyje është një porte permes së ciles klientet e HTTP mund të kërkojne serverin. Pasi është instaluar një nyje HTTP, mund të shtohen

store procedure-at dhe funksionet e përdoruesit për t'i bërë të vlefshme tek përdoruesit fundore. Kjo mund të ndodhe kur nyja është krijuar apo perditesuar. Kur proçedurat dhe funksionet janë të aktivizuara, ato specifikohen si metoda web-i. një koleksion i këtyre metodave që janë dizenjuar për tu përdorur bashke mund të quhet web service.



Figurë 33. Nyjet http në SQL Server 2008

Keto web service mund të përshkruhen duke përdorur formatin Web Services Description Language (WSDL).

Formati WSDL gjenerohet nga një instancë e SQL Server-it dhe i kthehet klienteve SOAP për çdo nyje HTTP mbi të cilen gjenerimi i WSDL është i aktivizuar, siç tregohet në Figurë 33. Nëse kërkohet, formati WSDL mund të jete një zgjidhje sipas deshires ndryshe nga ajo e gjeneruar nga SQL Server. Nyja mund të jete e konfiguruar në mënyrë të tillë që të mos i përgjigjet kërkesave të WSDL.

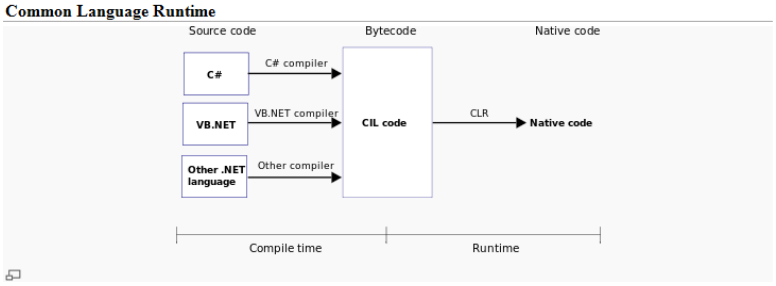
Një metodë web që është e lidhur me një nyje HTTP mund të jete një store procedure ose një skalar - valued user-defined function. Emri i metodës brenda SQL Server nuk është e detyrueshme të jete i njëjtë me emrin e metodës publike të webit. Implementimi i metodës web nuk ka ndonjë kod për të analizuar gramatikisht kërkesen apo të formatoje rezultatin që i kthehet konsumatorit të shërbimit, pasi SQL Server 2008 manaxhon të gjitha funksionalitetet. Nyjet suportojne identifikimin baze, identifikimin shifror, identifikimin e integrimit të Windows-it (NTLM or Kerberos), dhe identifikimin e serverit të SQL, por nuk suportojne kërkesat anonime. Kjo perzgjedhje e metodave identifikuese suporton thuajse çdo klient në një aplikacion SODA mixed-platform.

Expozimi i Web Services

Përdoret struktura create "endpoint t-sql" për të krijuar një nyje HTTP dhe ekspozohet një stored procedure si një metodë web.

6.2.3 Gjuha CLR dhe integrimi me SQL Server

Common Language Runtime (CLR) është një komponente e makines virtuale e .NET Framework e Microsoft-it dhe është pergjegjese për manaxhimin e ekzekutimit të programeve .NET. në një proçes të njohur si kompilimi në një kohë (just-in-time compilation), kodi i kompiluar konvertohet në instruksione makine që ekzekutohen nga CPU e kompjuterit. CLR siguron shërbime shtese duke perfshire manaxhimin e memories, sigurine e tipit, dhe manaxhimin e perjashtimeve. të gjitha programet të shkruara për .NET framework, pa marre parasysh gjuhen e programimit, ekzekutohen nga CLR. Kjo siguron manaxhimin e perjashtimeve, mbledhjen e "pleherave " dhe manaxhimin e thread-eve. CLR është njesoj në të gjitha versionet e .NET framework. CLR është implementimi i Sistemit Virtual të Ekzekutimit (VES – Virtual Execution System) të Microsoft-it percaktuar në standartet e Common Language Infrastrukture (CLI), fillimisht e zhvilluar nga vete Microsoft. Specifikimet e Common Language Infrastrukture percaktohen tashme nga një standard publik.



Figurë 34. Konvertimi nga CLR i CIL në kod native

[3]SQLCLR: Futja e Common Language Runtime (CLR) në berthamen e makines së bazës së të dhënave.

Integrimi i thelle është celesi i suksesit, një shtrese hostimi ndërmjet makines SQL dhe CLR lejon burimet, si psh memorjen dhe thread-et, të koordinohen ndërmjet kohës së ekzekutimit dhe makines së bazës së të dhënave.

Avantazhet e integrimit të CLR

Ne SQL Server 2005 dhe në versionet e mevoneshme të tij, objektet e bazës së të dhënave mund të krijohen me anë të CLR. Stored Procedure-at, Funksionet, Trigerat mund të kodohen në CLR. CLR është me e shpejte se T-SQL në shumë raste. CLR përdoret kryesisht për të plotesuar detyra të cilat nuk janë të mundura nga T-SQL ose që përdorin shumë burime. Implementimi i CLR siguron me tepër siguri për të zgjeruar Stored Procedure-at

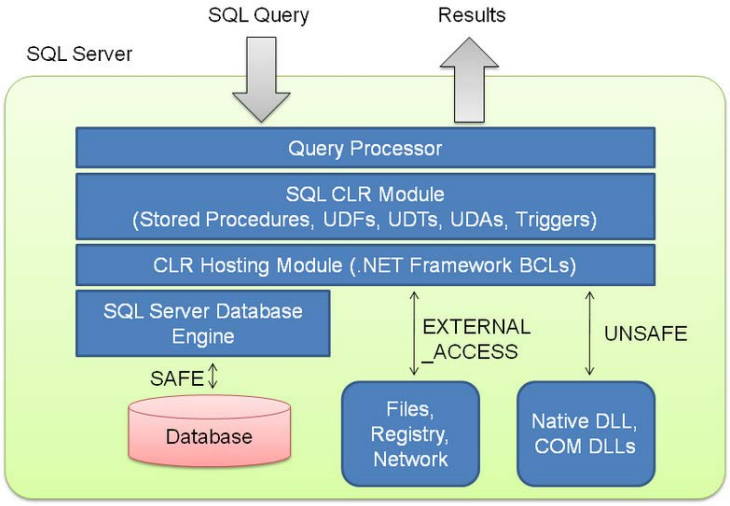
Ketij integrimi zakonisht i referohen si SQLCLR e cila është
Një framework që lidh mjedisin e makines së bazës së të dhënave SQL Server me ambjentin e programimit të .Net. Kjo lejon:
Prezantimi i integrimit të CLR, rriti së tepërmi mundësite e çfarë mund të bëhet brenda kontekstit të query-ve, Ato operacione që mund të behen me anë të T-SQL tashme

[3] http://en.wikipedia.org/wiki/Common_Language_Runtime

behen në mënyrë me efiçente dhe me lehtesisht në kod .Net. Ato operacione që nuk janë të mundura në T-SQL tashme janë të mundura të realizohen.

[4]Kërkesat e shërbimeve proçesohen brenda kufijve të implementimit të një shërbimi. Për vete sistemin e bazës së të dhënave që të jete një makine e besueshme SOA, duhet të hostoje logjiken e shërbimit në mënyrë sa me efiçente. Karakteristika SQLCLR integron Microsoft Common Language Runtime (CLR) brenda proçesit SQL Server runtime. CLR është një ambjent ekzekutimmi modern që suporton shumë gjuhe programimi si C#, Managed C++, Visual Basic, COBOL dhe të tjera.

Arkitektura SQLCLR



Figurë 35. Arkitektura SQLCLR

SQLCLR ndryshon në mënyrë radikale dhe zgjeron paradigmat e programimit të suportuara nga SQL Server 2008. Lejon zhvillimin e objekteve të mëposhtëme të bazës së të dhënave në një gjihe .NET:

- **Stored proçedurat e manaxhuara**: na sigurojnë mundësinë për të implementuar një stored procedure në një gjuhe .NET si C# ose VB. Gjithashtu permirëson performancën e logjikes së biznesit krahasuar me rastin kur është i shkruar në T-SQL, sepse T-SQL është një gjuhe programimi me të dhëna me të qendërzuara.

- **Funksionet e percaktuara nga përdoruesit**: me të cilat mund të shkruajme funksione vleresuese skalare apo tabelore direkt në kod .NET

- të **dhëna të percaktuara nga përdoruesit**: me një tip të dhënash të kastimizuar, jemi në gjendje të zgjerojme sistemin e tipit të te dhënave të SQL Server 2008.

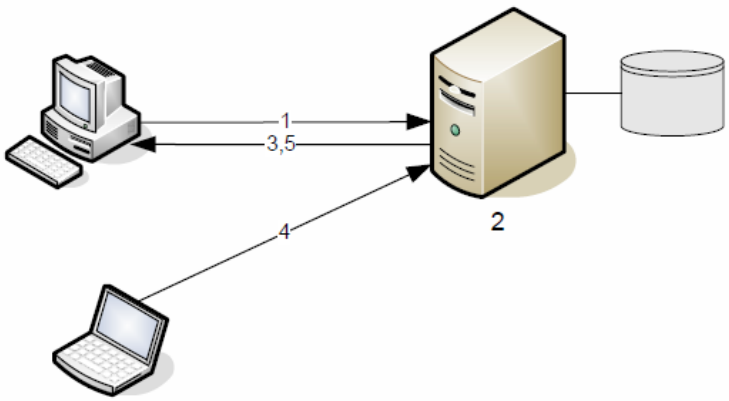
[4] http://www.sqlservercentral.com/articles/Stairway+Series/104406/

6.2.4 Database Change Notifications – njoftimet e ndryshimeve në bazën e të dhënave

[26] Shumë aplikacione të rëndësishme baze të dhënash ruajne në cache informacionin nga server-i i bazës së të dhënave për të shmangur aksesimin e bazës së të dhënave për çdo kërkese. këto cache shfaqin në shtresen e mesme të te dhënave të cache-se, cache-te e prezantimeve dinamike të permbajtjeve si dhe cache specifike të aplikacioneve të tjera. Ndërkohë që cache shkarkon serverin e bazës së të dhënave, ato duhet të balancojne si konsistencen e të dhënave dhe performancën edhe rregullat e rifreskimeve të pergjitheshme të cilat shpesh bazohen në rifreskim periodik nga back end.
Rifreskimi i cache-se në mënyrë shumë të shpeshte ngarkon bazën e të dhënave. Rifreskimi jo i shpeshte çon në gabime të aplikacioneve si pasoje e inkonsistences së të dhënave ndërmjet cache-se dhe të "te vertetes" së ruajtur në bazën e të dhënave. Njoftimet e ndryshimeve në bazën e të dhënave - **Database change notifications (DCN)**, lejon cache-ne klient për t'u regjistruar për të marre në formen e rezultateve të një query njoftimet kur një ndryshim në bazën e të dhënave mund të ndikoje në rezultatin e query-it të cache-se së regjistruar. Me anë të DCN, cache-ja klient shmang rifreskimin periodic dhe mund të reagoje shpejte ndaj ndryshimeve në bazën e të dhënave.

Workflow i pergjithshem për një cache që përdor DCN është si me poshtë:

1. një query SQL-je dërgohet tek serveri i bazës së të dhënave me një shenim të veçantë që tregon klientin që po bën kërkesen për njoftim mbi ndryshimet në baze të dhënash
2. një kërkese për regjistrim krijohet dhe regjistrohet për kete query. Query mbushet me informacion për të suportuar DCN dhe me pas optimizohet dhe ekzekutohet
3. Rezultatet e query-it i kthehen aplikacionit klient (cache-se).
4. Çdo ndryshim (DML apo DDL) që mund të ndikoje në rezultatin e query-it bën që makina e bazës së të dhënave të radhite një njoftim për ndryshime në aplikacionin e regjistruar.
5. Njoftimi për ndryshime ekzekutohet dhe dërgohet tek aplikacioni i regjistruar.



Figurë 36. Topologjia dhe rrjedha DCN

DCN-ja është e disponueshme në çdo klient që akseson të dhëna dhe është integruar në mënyrë transparente në mekanizmin cache të suportuar nga ASP.Net.

Kërkesat e njoftimeve

Një nga problemet klasike kur shkruhen aplikacionet me baze të dhënash të shpërndara është rifreskimi i të dhënave. Imgjinoni një website ose një aplikacion të zgjuar klient ku shfaqen lista e produkteve dhe cmimeve. Të njejtat të dhëna tërhiqen nga baza e të dhënave në mënyrë të vazhdueshme. Kjo është shumë joefiçente nëse të dhënat ndryshonin rralle, dhe do të shpërdoronte burimet dhe kohën e kezekutimit në serverin e bazës së të dhënave.
Ruajtja e të dhënave në cache është një teknikë për të minimizuar kërkesat e bera në bazën e të dhënave. të dhënat kërkohen vetëm një here dhe ruhen në cache në anen e klientit, dhe aplikacioni me pas në mënyrë të vazhdueshme akseson cache-ne. Me raste, cache-ja perditesohet për të rifreskuar të dhënat. Problemi në kete rast është kur do të përditesohet cache-ja. Nëse nuk e bëjmë shpesh, përdoruesit shohin të dhënat e vjetra, nëse e perditesojme cache-ne shumë shpesh, atehere reduktohet optimizimi në anen e serverit.
Për të zgjidhur kete problem përdoren query-t njoftuese. Ato na lejojne të ruajme të dhënat në cache dhe të njoftohemi nga server i SQL-se kur të dhënat ndryshojne. Me pas mund të bejme rifreskim të caches-se.
Ka dy arsye perse permenden query-t njoftuese tek SODA:

- **Query-t njoftuese nevojiten për SODA:**

ato janë teknologjia e katert e ofruar nga SQL Server 2008 për të suportuar SODA. Ato përdoren kryesisht në shtresen e mesme për të perditesuar cache-te kur ka ndryshime në serverin e bazës së të dhënave.

- **Query-t njoftuese bazohen në Service Broker**:

ato janë implementuar fillimisht me Service Broker. Sapo detektohet një ndryshim në të dhëna në serverin e bazës së të dhënave, query-t njoftuese gjenerojne një mesazh të ri Service Broker-I, i cili merr dhe dërgon tek i regjistruari që ka ruajtur të dhënat në cache. Nuk ka rendesi nëse i regjistruari është implementuar në shtresen e mesme si një web service apo si një klient i zgjjuar, pjesët e sigurta të Service Broker sigurohen që mesazhi dërgohet në mënyrë të suksesshme tek i regjistruari.

Kapitulli 7 – Arkitektura e Shërbimeve të Mesazheve dhe Radhëve

7.1 Shërbimet Broker të Mesazheve të bazuara në radhë në SQL Server

Sql Server 2008 Service broker është një framework mesazhesh asinkron që është e implementuar direkt në Sql Server 2008. Programet e bazuara në mesazhe na lejojne të implementojme software më fleksibel dhe të përshkallëzueshëm, ndryshe nga paradigmat e tjera të programimit si përshembull paradigmi klient-server.
gjatëproçesimit të mesazheve asinkron, dërguesi mund të vazhdoje punën ndërkohë që po pret nga marresi të proçesojë mesazhin dhe t'i kthejë përgjigje. Proçesimi i mesazheve asinkrone është i përdorshëm në rastet kur proçese biznesi të gjata thirren nga aplikacionet klient. Aplikacioni klient merr të dhënat e nevojshme nga përdoruesi, i transformon këto të dhëna në një mesazh dhe i çon këto të dhëna në një destinacion, ku kryhet dhe proçesimi në mënyrë asinkrone. Të mirat e kësaj ariktekture janë që aplikacionet klient janë me të shpejta dhe përdoruesi mund të beje gjera të tjera brenda aplikacionit ndërkohë që po proçesohet kërkesa e mepareshme. Kjo arkitekture mund të kërkoje disa destinacione për të implementur strukturen load-balancing, keshtu që kërkesat do të shperndahen dhe mund të kemi proçesim në paralel të mesazheve.
Kur implementojme zgjidhje të bazuara në mesazh, na duhet që proçeset të implementohen me paradigmin asinkron. Implementimi i zgjidhjeve me anë të mesazheve na çon të përdorim arkitektura jo tradicionale si klient server apo arkitektura me tre shtresa, por të shfrytezojme arkitektura që janë menduar për këto lloje aplikacionesh. Arkitekturat me të zakonshme janë (SOA): Service-oriented architecture dhe (SODA): Service-oriented database architecture.
SOA është një koleksion i disa modeleve arkitekturore me qëllim për të ndërtuar aplikacione heterogjene.
SODA është e njëjtë me SOA por këto modele janë të integruara direkt në databaze. E gjithe kjo teknologji bazohet në dialog dhe monolog. Dialogu është një bashkebisedim dy drejtimor egzaktesisht midis dy shërbimeve broker. Monologu është një bashkebisedim një drejtimor midis një proçesi server dhe një proçesi klient.
Tipet e mesazheve janë modele të gatshme ose të ndërtuara sipas rastit, ku tipi përcakton të dhënat që mesazhi do të permbaje. Çdo tipi mesazh duhet që t'i jete specifikuar dhe validimi në mënyrë që service broker të kontrolloje tipin e mesazhit. në këto momente service broker suporton këto tipe validimi: validim me anë të skemave të XML, validim me anë të "well formed XML", pa validim (për të dhënat binare) dhe bosh(për mesazhe që janë pa permbajtie).

Rradhët në service broker janë memorie të dedikuara për mesazhet e marra që presin për t'u proçesuar. një rradhë duhet të percaktohet nga shërbimi inicializues dhe shërbimi qëllim. Kur vjen një mesazh i ri sistemi e validon ate, dhe pas validimit e fut ate në rradhë për të pritur për ekzekutim. Rradhët janë implementuar me anë të tabelave të fshehura që janë futur për here të pare në SQL Server 2005. Rradhët janë tabela të zakonshme të ruajtura në memorje, por ndaj tyre nuk mund të përdoren rregulat e SQL

për të manipuluar të dhënat. Me anë të SQl, vetëm mund të aplikohen SELECT ose e thënë ndryshe kemi vetëm të drejta për lexim.
Programet e shërbimit në service broker janë komponente që ruajne një procedure dhe e aktivizojne ate kur një mesazh vjen në rradhë për të ekzekutuar. këto komponente mund të aktivizohen automatikisht nga sistemi, mund të skedulohen që në kohë të caktuara të ekzekutohen programe të caktuara ose mund të aktivizohen në baze të një kushti logjik.
Service broker lejon SQl servera të ndryshem të komunikojne ndërmjet tyre në mënyrë të sigurt. Aplikohen dy tipe të ndryshme sigurie: siguria në transport dhe siguria në dialog.

- **Siguria në transport** ndalon SQL servera të pa autorizuar t'i cojne mesazhe service brokerit.
- **Siguria në dialog** siguron metoda kriptimi për komunikim dhe verifikon indetitet e SQL serverave që po marrin pjesë në dialog.

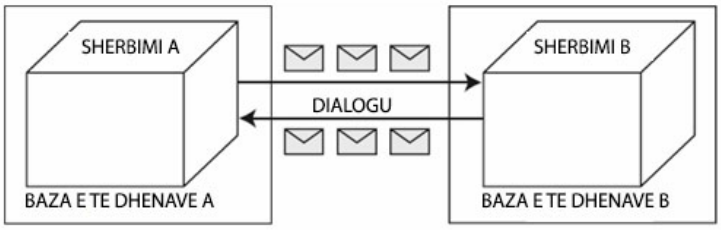
Performanca është një kusht i rëndësishem në të gjithe sistemet software. Service broker ofron performancë në disa fusha, pershembull në logjiken e proçesimit të mesazheve, rradhë proçesimi të shumta, menaxhim të trasanksioneve, dhe single log writers. Disa të mira të service brokerit janë: përdorimi i një API të vetëm, instrumente për administrim të centralizuar, mesazhet e besuara dhe është i përshkallëzueshem dhe i mbyllur në baze të skenareve.

7.2 Arkitektura Service Broker

Komponentet kryesore të arkitektures së shërbimit Broker janë si me poshtë :

• **Komunikimi**: një komunik është i besueshem, i urdheruar për të shkembyer mesazhe ndërmjet dy shërbimeve Service Broker. Arkitektura e Service Broker përcakton dy lloje komunikimi:

• **Dialogu**: është një komunikim dy drejtimor ndërmjet ekzaktesisht dy shërbimeve Service Broker. Shërbimet ekzistojne në të dyja anet e dialogut, të marresit dhe dërguesit. Shërbimit në anen e dërguesit, i referohemi si shërbimi iniciator. Shërbimi iniciator fillon një dialog të rid he dërgon mesazhin e pare tek shërbimi i targetuar. të dyja shërbimet mund të shkëmbëjnë me pas mesazhe në të dyja drejtimet.
• **Monologu**: një monolog është një komunikim nje-drejtimor ndërmjet një shërbimi të vetëm publikues dhe disa shërbimeve të regjistruar. Ky është një version i besueshem i paradigmes publikues-regjistrues. Aktualisht, monologjet nuk suportohen në Service Broker, por ato mund të jenë të perfshira në një version të ardhshem të Service Broker.



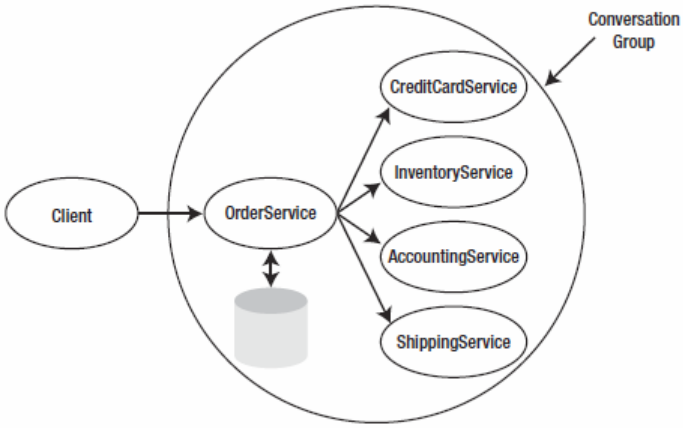
Figurë 37. Një dialog i shërbimeve broker

Grupet e bashkëbisedimit (Conversation Groups)

Një grup bashkebisedimi identifikon një ose me shumë biseda relevante dhe lejon një aplikacion Service Broker të koordinoje lehtesisht bisedat të perfshira në një detyre të caktuar biznesi. Çdo bisede i perket një grupi bashkebisedimi, dhe çdo grup bashkebisedimi është i lidhur me disa biseda nga shërbime të ndryshme.
Për shembull, kur porositim libra online, shërbimi i hyrjes së porosise dërgon mesazhe në disa shërbime të tjera dhe fillon një proçess biznesi për një kohë të gjate. Shërbimet e tjera mund të jete një ose të gjitha me poshtë:

- Shërbimi i validimit të kartes së kreditit
- Shërbimi i inventarit
- Shërbimi i kontabilizimit
- Shërbimi i dergimit

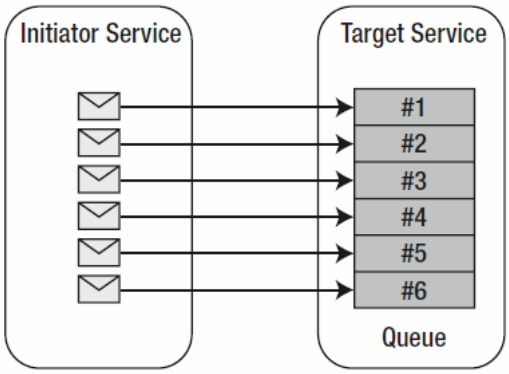
Shërbimi i futjes së porosise fillon kater dialogje të ndryshme në çdo prej këtyre dialogjeve individual. SB i grupon këto kater dialogje sebashku në një grup bashkebisedimi. këto kater shërbime mund të përgjigjen të gjitha thuajse në të njëjtën kohe, keshtu që është e mundur që mesazhet përgjigjes për të njëjtën porosi mund të proçesohen në thread-e të ndryshme madje dhe pa qene të ndergjegjshme për njëra-tjetrën. Për të zgjidhur kete problem, SB kyç grupet e bashkebisedimit.



Figurë 38. Grupet e Komunikimit

Sekuencimi i Mesazheve

[24] në Service Broker, sekuencimi dhe udhezimi i mesazheve sigurohen në kontekstin e gjithe jetes së një komunikimi. Sekuencimi dhe udhezimi i mesazheve mirëmbahen nëpërmjet numrave sekuencial, të cilet inkrementohen për çdo mesazh të dërguar.



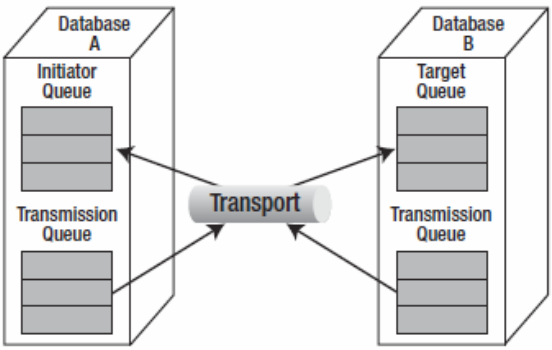
Figurë 39. Renditja e Mesazheve në SB

Dergim i besueshem i mesazheve

Service Broker siguron gjithashtu një dergim të besueshem të mesazheve brenda kontekstit të një dialogu.

Service Broker nuk mund të siguroje që mesazhet të merren në renditjen e dërguar as që të gjitha mesazhet të merren. Kur mesazhet humbasin gjatëproçesit të dergimit, sekuenca e dergimit të mesazheve permban hapesira boshe, të cilat nuk janë të përshtatshme në dialog. Service Broker bën e mundur dergimin e mesazheve të besueshme, sepse dërguesi i ridërgon mesazhet e humbura në mënyrë periodike deri sa merr një njoftim nga marresi për mesazhet e marra.

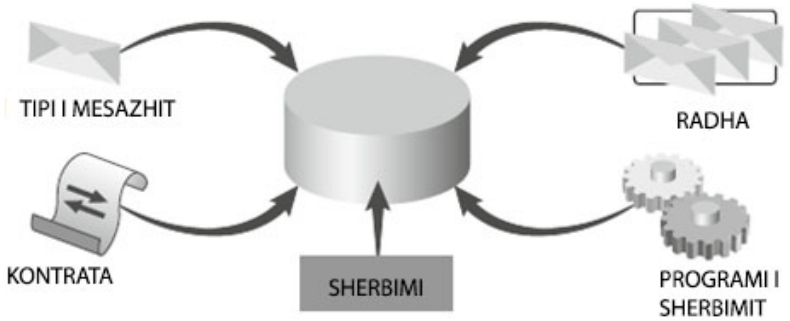
Protokolli i ridergimit dhe i njoftimit është i ndërtuar direkt brenda infrastrukturës së Service Broker, keshtu që është totalisht transparent për zhvilluesit e aplikacionit.



Figurë 40. Proçesi i besueshem i mesazheve në SB

Në figurën 40, mund të shihen që mesazhet që janë dërguar permes rrjetit, vendosen në një rradhë të perkohëshme të quajtur rradha e transmetimit.
Service Broker dërgon mesazhe në rrjet dhe i markon si "ne pritje" për identifikim në kete rradhë transmetimi. Kur një mesazh merret nga shërbimi destinacion dhe ruhet në radhen e targetuar për proçesim të metejshem, marresi dërgon një njoftim mbrapsh tek dërguesi – tek iniciatori i shërbimit. Kur dërguesi e merr kete mesazh njoftimi, ai fshin mesazhin nga radha e transetimit. Shërbimi inicues duhet të përcaktojë gjithmone një radhe. Kjo radhe përdoret për dy qëllime. Qëllimi i pare është kur shërbimi target deshiron të dërgoje një përgjigje mbrapsh tek shërbimi inicues. Qëllimi i dyte është për manaxhimin e gabimeve. Shërbimi target duhet të jete në gjendje të dërgoje një mesazh gabimi mbrapsh tek shërbimi inicues. Mesazhi ruhet në radhen e iniciatorit.

Koncepti kryesor në SB është një shërbim i cili është i përbërë nga elementet tipi mesazhit, kontrata, radha dhe shërbimi të ilustruar në figurën 41 më poshtë.



Figurë 41. Objektet e shërbimit SB

[30]Tipi mesazhit (Message Types) janë modele të gatshme ose të ndërtuara sipas rastit, ku tipi përcakton të dhënat që mesazhi do të përmbajë, emrin përkatës si dhe tipin e kontrollit që do kryhet në një mesazh. Tipi i mesazhit duhet të krijohet në çdo databazë që do të marrë pjesë në komunikim.

Kontrata(Contract) përfaqëson një marrëveshje për mesazhet që do të dërgohen midis dy shërbimeve SB si dhe përcakton se në cilën pjesë të bashkëbisedimit mund të dërgohet një tip i caktuar mesazhi.

Radhët (Queue) në SB janë hapësira memorje të dedikuara për mesazhet e marra pas proçesit të kontrollit që presin për tu proçesuar. Rradhët duhet të krijohen si për shërbimet dërguese ashtu dhe për ato marrëse. Ato implementohen në Sql Server me anë të tabelave të fshehura tek të cilat nuk mund të bëjmë modifikime por vetëm lexim.

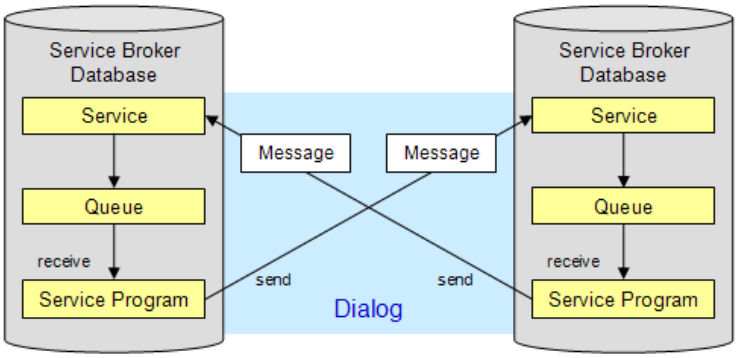
Shërbimet (Service Program) në SB janë komponente të cilat mund të jenë në trajtën e një proçedure në rast aktivizimi në brendësi të databazës apo dhe në formë programi kur aktivizohet nga jashtë saj. Komponentet mund të aktivizohen automatikisht nga sistemi, mund të skedulohen ose mund të aktivizohen në bazë të një kushti llogjik.

Proçesimi i Objekteve Service Broker

Si permbledhje mund të themi që objektet kryesore të shërbimeve broker në bazën e të dhënave janë : mesazhet , kontratat e shërbimeve, radhet , shërbimet, dialogu, grupet e bashkebisedimit dhe programet shërbim.

Ndërtimi i një ofruesi shërbimesh perfshin disa hapa për të krijuar dhe konfiguruar disa ose të gjitha këto objekte, duke përdorur forma të reja të ndryshme të thirrjeve të Transact-SQL CREATE. Sapo këto objekte vendosen, mund të dërgohen mesazhe në format XML duke përdorur thirrjen Transact-SQL SEND dhe një program shërbimesh mund ti terheq ato nga rradha e marresit me një thirrje RECEIVE.

Mesazhet dërgohen nën shërbimet që përcaktohen dhe më pas mesazhet vendosen në një rradhë pritjeje për tu proçesuar nga një program shërbimi, sic tregohet në figurën 42. Me shpesh një program shërbimi është një stored procedure që sillet pak a shumë Windows message pump, që proçeson mesazhet dhe dërgon përgjigjet. SB përdor një model që është diçka midis modeleve të terheqjes dhe ngjarjeve, duke aktivizuar programin e shërbimit që kufizohet me një rradhë të nevojshme për proçesimin e mesazheve që kanë mberritur. SB mund të aktivizoje programe shtese shërbimesh deri në një limit të paracaktuar të nevojshme për të shmangur fenomenin e quajtur "gryka e shishes" dhe të manaxhoje në mënyrë efektive 'peak loads'



Figurë 42. Struktura e shërbimit Broker dhe sinjali i dialogut

Rradha e SB është kryesorja në aspektin e besueshmerise së suportit, të proçesimit të mesazheve asinkrone. Për shkak se rradha është një tabelë baze të dhënash, mesazhet e rradhitura marrin të njejtin benefit si dhe të dhënat e tjera relacionale, duke perfshire recovery në rastet kur baza e të dhënave deshton dhe garanton dergimin në rradhë të mesazheve duke përdorur transaksione me të njejtat benefite ACID (pandashmeri, konsistence, izolim, jetegjatesi) si të dhënat e tjera. SB bën me shumë se thjeshte manaxhimin e rradhës. Ai ofron një framework të kompletuar për të suportuar aplikacione që kërkojne proçess të besueshem të shkembimit të mesazheve. Pervec shpetimit të zhvilluesve nga krijimi i infrastrukturës komplekse të proçesit të shkembimit të mesazheve – një detyre e tmerrshme – SB manaxhon mesazhet në mënyrë të tillë që ato të dërgohen në renditjen e duhur. Ajo gjithashtu manaxhon biseda, kontrata, lidhjet e shebimeve të rutimit dhe të shërbimeve remote.

7.3 Ndërtimi i aplikacioneve i bazuar në radhën e mesazheve duke përdorur shërbimet Broker në Sql Server.

[25]Shkëmbimi i të dhënave ndërmjet aplikacioneve të ndryshme të cilat përdorin databaza të ndara kategorizohet si një operacion i komplikuar, me kosto dhe ndonjëherë me risk për biznesin. Një ndër strategjitë më të sukseshme për të bëre të mundur këte shkëmbim është implementimi në brendësi të databazës i arkitekturës së orientuar drejt shërbimeve (SOA), të cilat lejojnë aplikacione të ndryshme të shkëmbejnë të dhëna me njëra tjetrën ndërkohë që ndërveprojnë me proçeset e biznesit.SOA është një studim në zhvillim në të cilin përfshihen gjithë kërkesat për shërbime të pavaruara, të bazuara në standarte dhe protokolle të pavaruara në platformat e shpërndara, në mënyrë që të rritet efektshmëria për zhvilluesit që të krijojnë aplikacione të sofistikuara. Në këtë artikull do të shpjegoj se si mund të integrojmë principet e SOA në databazën SQL Server 2008 me anë të Service Broker(SB). SB në Microsoft SQL Server 2008 (SSB) është një metodologji e cila mundëson funksionin e një radhe të mesazheve midis instancave të ndryshme në databaza.

Arkitektura e orientuar drejt shërbimeve (SOA) është një paradigmë për organizimin dhe shfrytëzimin e aplikacioneve të shpërndarë në rrjet, që mund të jenë nën kontrollin e fushave të ndryshme duke përdorur implementimin e teknologjive të ndryshme të cilat kanë si karakteristikë krijimin e shërbimeve të pavarura dhe ripërdorimin e tyre. SOA promovon bashkimin e lirë midis komponenteve software në mënyrë që ato mund të ripërdoren. Aplikimet SOA ndërtohen në bazë të shërbimeve të cilat janë zbatimi i një funksionaliteti të përcaktuar mirë të biznesit. Në vazhdimësi shërbime të tilla mund të konsumohen nga klientët në aplikime të ndryshme apo nga proçese të biznesit. SOA lejon ripërdorimin e aseteve ekzistuese, ku shërbimet e reja mund të krijohen nga një infrastrukturë ekzistuese e sistemeve IT.
Vitet e fundit është vënë re një përshtatje në shkallë të gjerë e SOA me biznesin dhe si rrjedhojë shumica e organizatave të mëdha kanë reduktuar kostot, rreziqet duke bërë investime në përshtatjen e shërbimeve të pavarura në sistemet IT.
Modeli SOA mund të implementohet direkt në nivel databaze dhe ndryshe i referohemi me Service Oriented Database Architecture (SODA).

SQL Server Service Broker (SSB)
Microsoft SQL Server mundëson implementimin e SOA nëpërmjet Service Broker. SSB është një framework mesazhesh asinkron që është i implementuar direkt në databazën SQL Server 2008. Dërguesi i mesazhit mund të vazhdojë punën ndërkohë që po pret nga marrësi të proçesojë mesazhin dhe ti kthejë përgjigje. Proçesimi i mesazheve asinkronë është i përdorshëm në rastet kur proçese biznesi të gjata thërriten nga aplikacionet klient. Aplikacioni klient merr të dhënat e nevojshme nga përdoruesi, i transformon këto të dhëna në një mesazh dhe i dërgon në një destinacion, ku kryhet dhe proçesimi në mënyrë asinkrone. Të mirat e kësaj arkitekture janë që performanca e aplikacioneve klient është më e lartë dhe përdoruesi mund të bëjë gjëra të tjera brenda aplikacionit ndërkohë që po proçesohen kërkesat e dërguara.

Metodologjia

Në SQL Server 2008 do të krijohet një tabelë në të cilën do të hidhen të gjitha transaksionet, të cilat mund të jenë rezultat i veprimeve të databazës si insert, update, delete. Duke u nisur nga fakti që veprimet në këtë tabelë janë të shumta dhe të nisura nga burime të ndryshme, ekziston një kërkesë nga biznesi e cila lidhet me krijimin e një tabelë historike log në të cilën duhet të arshivohen disa nga karakteristikat kryesore të veprimit si data, përdoruesi, tipi i veprimit. Teknikisht këtë kërkesë do ta zbatojmë duke përdorur shërbimet direkt në SQL Server 2008 të mundësuara nga SB dhe në përfundim do të bëjmë një krahasim me zgjidhjet e tjera alternative.

Materialet dhe mjetet e përdorura

Në versionin SQL Server 2008 R2 Enterprise Edition është krijuar një databazë për ambjentin ku do të punohet me emrin SBB2. Si editor query është përdorur SQL Server 2008 Management Studio. Gjuha për krijimin e objekteve dhe komponenteve të SB është T-SQL.

Krijimi i komponenteve në SSB

Fillimisht aktivizojmë shërbimin Broker në databazën SBB2.

```
ALTER DATABASE SBB2 SET ENABLE_BROKER;
```

Krijojmë dy tipet e mesazheve dërguese dhe marrëse të cilat do të kontrollohen sipas rregullave të formatit well_formed_XML.

```
ALTER DATABASE SBB2 SET ENABLE_BROKER;
```

Krijojmë dy tipet e mesazheve dërguese dhe marrëse të cilat do të kontrollohen sipas rregullave të formatit well_formed_XML.

```
CREATEMESSAGETYPE
    [//DB/SBB2/RequestMessage]
VALIDATION=WELL_FORMED_XML;
CREATE MESSAGETYPE
    [//DB/SBB2/ReplyMessage]
VALIDATION=WELL_FORMED_XML;
```

Krijojmë kontratën Kontrata_SB, e cila do të përcaktojë komunikimin midis dy shërbimeve.

```
CREATE CONTRACT [//DB/SBB2/Kontrata_SB]
```

([//DB/SBB2/RequestMessage]

SENT BY INITIATOR,

[//DB/SBB2/ReplyMessage]

SENT BY TARGET)

Në SSB krijojmë dy radhë burim dhe destinacion, të cilat do të shërbejnë si hapësire memorje përkatësisht për të gjitha mesazhet që do të dërgohen dhe ato që do të merren në destinacion.

CREATE QUEUE Radhë_Burim

CREATE QUEUE Radhë_Destinacion

Krijojmë dy shërbimet iniciues dhe target të cilat do të proçesojnë pikërisht mesazhet që do të gjenden në radhët e krijuara më lart.

CREATE SERVICE

[//DB/SBB2/InitiatorService]

ON QUEUE Radhë_Burim;

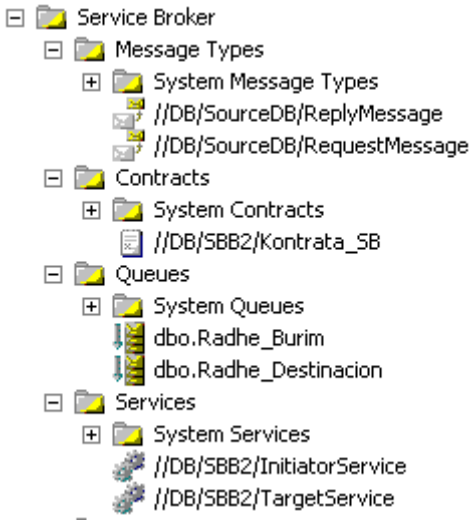
CREATE SERVICE

[//DB/SBB2/TargetService]

ON QUEUE Radhe_Destinacion

([//DB/SBB2/Kontrata_SB]);

Një pamje e komponenteve të krijuara në SSB do të ishte si në figurën 43.



Figurë 43. Objektet SB të krijuara në SQL Server 2008

Krijojmë dy tabelat Transaksione, Transaksione_Log në të cilat do të hidhen përkatësisht rreshtat e transaksioneve dhe historiku i ndryshimeve.

Pas krijimit të komponenteve kryesore të SB, një ndër mënyrat për të filluar shkëmbimin e mesazheve është përdorimi i trigerave në SQL Server.Pjesë e kodit triger e cila fillon dialogun dhe dërgon mesazhet në destinacion do të ishte.

```
SET @Transaksion_Id =NEWID()
BEGIN DIALOG @InitDlgHandle FROM SERVICE     [//DB/SBB2/InitiatorService]
TO SERVICE  N'//DB/SBB2/TargetService'
ON CONTRACT
    [//DB/SBB2/Kontrata_SB]
WITH
ENCRYPTION=OFF;
SET @RequestMsg =
        (SELECT sender.*FROM
(
SELECT @Transaksion_Id as Id,'Inserted' as veprimi,* FROM inserted
) sender FORXMLAUTO);
IF @RequestMsg isnotnull
BEGIN
;SEND ON CONVERSATION @InitDlgHandle
MESSAGETYPE
    [//DB/SBB2/RequestMessage]
(@RequestMsg);
END
```

Programi shërbim mund të aktivizohet me anë të një proçedure SB_ReceiveData në brendësi të databazës e cila do të jetë gjithë kohës në egzekutim. Pjesa e kodit të kësaj proçedure e cila do të shtonte rreshtat në arshivën log jepet më poshtë.

```
IF @RecvReqMsg ISNOTNULL
BEGIN
INSERT INTO Transaksione_Log(XMLHistory)
SELECT @RecvReqMsg
END
COMMIT TRANSACTION;
```

```
END END GO
```

Rezultatet

Për të parë rezultatet e konfigurimit të SB në SQL server si dhe arshivën log zbatojmë hapat e mëposhtëm.
Mbi tabelën transaksione kryejmë disa veprime insert, delete duke përdorur gjuhën T-SQL.

```
WHILE @i > 0
BEGIN
  SELECT @i = @i - 1
  INSERT INTO Transaksione SELECT @i,'Insert',Getdate()
  DELETE FROM Transaksione where transaksion_id= @i
END
```

Pas këtij momenti, mesazhet janë iniciuar në mënyre asinkrone nga radha burim dhe pas kontrollit janë dërguar në radhën destinacion. Për të parë mesazhet në radhën destinacion, egzekutojmë komandën si më poshtë

```
SELECT top 4 queuing_order, casted_message_body =
CASE message_type_name WHEN 'X' THEN CAST(message_body AS
NVARCHAR(MAX))
 ELSE message_body
END
FROM [SSB4].[dbo].[Radhe_Destinacion] WITH(NOLOCK)
```

Results | Messages

	queuing_order	casted_message_body
1	40	<sender Id="4AA92F6E-F329-42A6-9239-3EB625999DCC" veprimi="Inserted" Transaksion_Id="4" Transaksion_Tipi...
2	41	<sender Id="C4A0281F-114B-435D-B32D-A1425CEF7643" veprimi="Deleted" Transaksion_Id="4" Transaksion_Tipi...
3	42	<sender Id="81BA226F-6A9A-49A5-B88D-43B7B1E355BF" veprimi="Inserted" Transaksion_Id="3" Transaksion_Tip...
4	43	<sender Id="C13C13AA-04FC-48F7-ADEE-029453BDD26D" veprimi="Deleted" Transaksion_Id="3" Transaksion_Ti...

Figurë 44. Outputi i radhës Destinacion

Në këtë moment tabela historike transaksione_log nuk ka rreshta për arsyen që nuk kemi të aktivizuar programin shërbim. Kjo bëhet e mundur me egzekutimin e proçedurës **SB_ReceiveData**. Pas këtij momenti radha destinacion boshatiset dhe në tabelën **transaksione_log** hidhen të gjithë mesazhet që po prisnin në radhë.

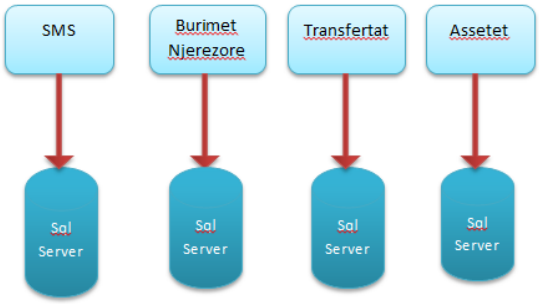
Zgjidhje alternative e kësaj kërkese do të ishte përdorimi i arshivës log të vetë databazës dhe në këtë rast do të përballeshim me situatën e logut shumë të madh, një

format i vështirë për tu përshtatur me kërkesat e biznesit dhe një mirëmbajtje me kujdes të veçantë në rast të veprimeve si Shrink,Backup Log etj.
Një tjetër zgjidhje më praktike do të ishte implementimi në nivel aplikacioni jashtë databazës për të marrë informacionin nëpërmjet trigerave dhe për ta arshivuar në tabelën historike, por e krahasuar me zgjidhjen e ofruar nëpërmjet shërbimeve SB, performanca e aplikacionit në fjalë do të ishte më e ulët në rast se sasia e ndryshimeve do të ishte e madhe apo edhe nëqoftëse në tabelën log do të duhej të arshivoheshin ndryshimet dhe nga tabela të tjera.Kjo pasi mesazhet me SB dërgohen në trajtë asinkrone, gjë e cila zvogëlon kohën e pritjes së aplikacionit.Në këte variant, ndryshime në llogjikën e arshivimit kanë nevojë për një mirëmbajte shtesë të kodit të aplikacionit.

Gjithashtu performaca rritet ndjeshëm kur zgjidhja e bazuar në shërbimet SB mund të kërkojë disa destinacione për të implementur strukturën load-balancing, kështu që kërkesat do të shpërndahen dhe do të kemi proçesim në paralel të mesazheve.
Një variant akoma më i përmirësuar i zgjidhjes së ofruar do të ishte përdorimi i teknikës change tracking në SQL Server bashkë me SSB, metodë e cila kombinon teknikën më të mirë për monitorimin e ndryshimeve në databazë së bashku me shërbimet në SSB.Ky kombinim do të ilustronte më dukshëm pavarësinë e aplikacionit nga ndryshime të mundshme të skemës në databazë.

Duke u nisur nga diskutimet me lart do të trajtoj rastin e optimizimit të një sistemi auditimi të gjitha ndryshimeve nga përdoruesit për aplikacionet në një sistem bankar.

Në një sistem bankar përveç aplikacionit kryesor bankar, egzistojnë të tjera aplikacione kryesore që përdoren ku disa prej tyre mirëmbahen nga palë të treta, të cilat ndodhen në servera të ndryshëm në rrjet dhe që përdorin baza të dhënash të ndryshme. Ndër aplikacionet e shumta për të trajtuar problemin në eksperiment do të përzgjedh këto sisteme
-Aplikacioni kryesor transaksional në bankë
-HR, Burimet Njerëzore
-Programi i menaxhimit të aseteve
-Sistemi i dërgimit të mesazheve me celular



Figurë 45. Disa prej aplikacioneve kryesore ne nje sistem bankar

Problemi

Një ndër rekomandimet me prioritet të lartë nga auditorët është :

- Për secilin prej aplikacioneve kryesore duhet të egzistojë ose të implementohet një modul, në mënyrë që të regjistrohen të gjitha gjurmët e transaksioneve kryesore
- Këto veprime të kryera së bashku me datat mund të identifikohen për ç'do përdorues

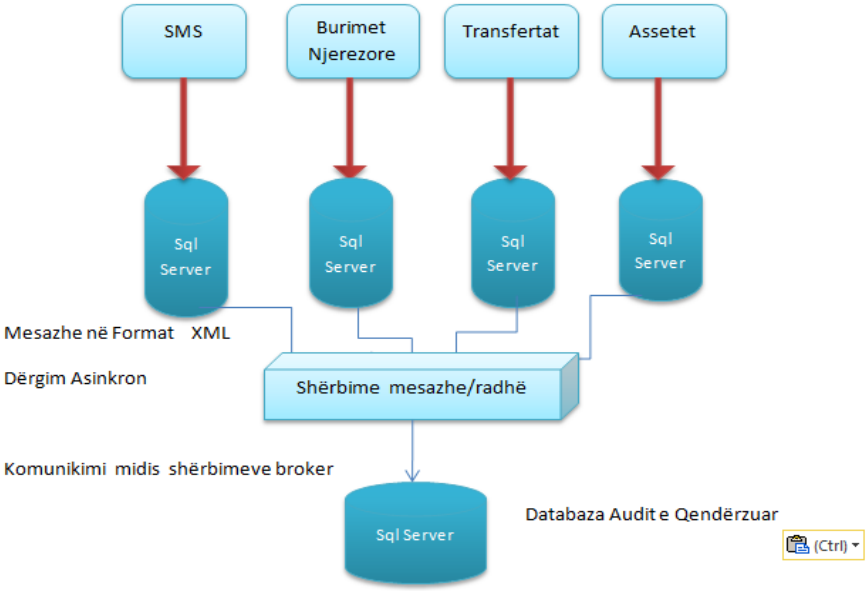
Për të trajtuar rekomandimin, zgjidhja e parë që listohet është pikërisht diskutimi me kompanitë që mirëmbajnë këto aplikacione apo me pale të treta që mund të ofrojnë zgjidhje për këtë rekomandim.
Nga kjo zgjidhje evidentohen këto probleme:

- *Ndryshimi në logjikën e programimit i të gjitha aplikacioneve*
- *Kohëzgjatja për aplikimin, pasi moduli shtesë për ç'do aplikacion supozohet si një ndryshim kryesor në sistem dhe do të kalojë në të gjitha fazat e një "change management"*
- *Ndikimi i drejtëpërdrejtë në rritjen e madhe të kostos*
- *Degradimi i performancës së aplikacioneve egzistuese. Gjurmët për ç'do aktivitet do të rrisin kohën e përfundimit të veprimeve.*
- *Do të përballemi me mirëmbajte të logut të aktivitetit për ç'do aplikacion, si pjesë lokale e databazës.*
- *Në kushte të veçanta të infrastrukturës së rrjetit ulet ndjeshëm performanca*

Zgjidhja e propozuar

- *Implementimi i një shtrese ndërmjetëse në këtë mjedis të shpërndarë bazash të dhënash me anë të shërbimeve, e cila do të shërbejë për dërgimin e gjurmëve të ç'do aktiviteti në format XML*
- *Zgjidhja do të implementohet direkt në bazën e të dhënave SQL server, i cili mundëson krijimin e aplikacioneve bazuar në mesazhet e besueshëm dhe radheve.Kjo zgjidh problemin e ndryshimit të llogjikës aktuale të programimit të aplikacioneve të veçanta.*
- *Si mjet do të përdoren shërbimet broker të trajtuara si në [25], të cilat rrisin ndjeshëm performancën e aplikacionit pasi dërgojnë mesazhe asinkron sipas një radhe të caktuar*
- *Do të krahasohet performanca me dhe pa përdorimin e shërbimeve broker*
- *Për të zgjidhur problemin e menaxhimit për ç'do databazë si pasojë e rritjes nga aktiviteti,të gjitha gjurmët do të arshivohen në një databazë të qëndërzuar dhe në këtë sistem të dhënash me kapacitete të larta do të optimizohet duke përdorur teknikat e*
  - *-Particionimit për të optimizuar kërkesat në këtë sistem të qëndërzuar*
  - *-Arshivimit duke përdorur ndarjen në filegroups*
  - *-Kompresimit për të menaxhuar rritjen e pa-kontrolluar të të dhënave*

Arkitektura e propozuar



Figurë 46. Sistemi audit i qendërzuar

Duke u bazuar në sintaksën e krijimit të konfigurimit të Service broker, kontratave, radhëve, mesazheve dhe shërbimeve, më poshtë jepet kodi me anë të gjuhës t-sql për bazën e të dhënave të burimeve njerëzore(HR) dhe databazës qendrore Audit.

Kodi në databazën DB_AUDIT

```
USE master
ALTER DATABASE DB_AUDIT SET ENABLE_BROKER
GO
USE DB_AUDIT
GO

CREATE TABLE dbo.Transaksione_Audit
(
	Id BIGINT IDENTITY(1,1),
	Makina_Burim sysname NOT NULL,
	DB_Burim sysname NOT NULL,
	Tabela_Burim sysname NOT NULL,
	UserID NVARCHAR(500) NOT NULL,
	Veprimi varchar(10) NOT NULL CHECK (Veprimi IN ('Fshij', 'Modifikim',
'Shtim')),
	TeDhenat XML NOT NULL,
	Data DATETIME NOT NULL DEFAULT GETUTCDATE()
)

GO

CREATE PROCEDURE dbo.usp_shkruaj_tedhena
AS
BEGIN
	DECLARE @mesazhi XML
	DECLARE @dialogu uniqueidentifier
```

```
WHILE(1=1)
BEGIN
	BEGIN TRANSACTION
	BEGIN TRY
	;RECEIVE top(1)
				@mesazhi	= message_body,
				@dialogu		= conversation_handle
		FROM  dbo.radha_Destinacion

	  IF @@ROWCOUNT = 0
		BEGIN
			IF @@TRANCOUNT > 0
			BEGIN
				ROLLBACK;
			END
			BREAK;
		END

		SELECT @mesazhi, @dialogu

		DECLARE @Makina_Burim sysname, @DB_Burim sysname,
@Tabela_Burim sysname,
				@UserID NVARCHAR(500), @Veprimi CHAR(1),
@TeDhenat XML


		SELECT		@Makina_Burim =
T.c.query('/AuditMsg/Makina_Burim').value('.[1]', 'sysname'),
				@DB_Burim =
T.c.query('/AuditMsg/DB_Burim').value('.[1]', 'sysname'),
				@Tabela_Burim =
T.c.query('/AuditMsg/Tabela_Burim').value('.[1]', 'sysname'),
				@UserID =
T.c.query('/AuditMsg/UserId').value('.[1]', 'NVARCHAR(50)'),
				@Veprimi =
T.c.query('/AuditMsg/Veprimi').value('.[1]', 'CHAR(1)'),
				@TeDhenat = T.c.query('*')
		FROM  @mesazhi.nodes('/AuditMsg/TeDhenat') T(c)

		INSERT INTO dbo.Transaksione_Audit(Makina_Burim, DB_Burim,
Tabela_Burim, UserID, Veprimi, TeDhenat)
		SELECT @Makina_Burim, @DB_Burim, @Tabela_Burim,
@UserID, @Veprimi, @TeDhenat

		IF @@TRANCOUNT > 0
		BEGIN
			COMMIT;
		END
	END TRY
	BEGIN CATCH
		IF @@TRANCOUNT > 0
		BEGIN
			ROLLBACK;
		END
```

```
                END CATCH;
        END
END

GO

CREATE MESSAGE TYPE [//Audit/mesazhi]
        VALIDATION = WELL_FORMED_XML


CREATE contract [//Audit/kontrata]
        ([//Audit/mesazhi] SENT BY INITIATOR)

CREATE QUEUE dbo.radha_Destinacion
        WITH STATUS = ON,
        ACTIVATION (
                PROCEDURE_NAME = usp_shkruaj_tedhena,
                MAX_QUEUE_READERS = 100,
                EXECUTE AS 'dbo' );


CREATE SERVICE [//Audit/shkruaj_tedhena]
        AUTHORIZATION dbo
        ON QUEUE dbo.radha_Destinacion ([//Audit/kontrata])

GRANT SEND ON SERVICE::[//Audit/shkruaj_tedhena] TO PUBLIC
GO

CREATE ROUTE [RouteDataReceiver]
        WITH ADDRESS = N'TRANSPORT'
```

Kodi në databazën HR (i njëjti kod)

```
SE HR_DB

CREATE ROUTE RouteDataSender
        AUTHORIZATION dbo
WITH
        SERVICE_NAME = '//Audit/shkruaj_tedhena',
        BROKER_INSTANCE = '6A028869-F0C2-4CE5-95AC-2652CD0B4305',
        ADDRESS = 'TCP://10.0.0.116:4022'
GO

CREATE MESSAGE TYPE [//Audit/mesazhi]
        VALIDATION = WELL_FORMED_XML


CREATE contract [//Audit/kontrata]
        ([//Audit/mesazhi] SENT BY INITIATOR)

CREATE QUEUE dbo.Radha_Inicues

CREATE SERVICE [tcp://10.0.0.113:4022/HR_DB/Audit/derguesi]
        AUTHORIZATION dbo
```

```
        ON QUEUE dbo.Radha_Inicues

GRANT SEND ON SERVICE::[tcp://10.0.0.113:4022/HR_DB/Audit/derguesi] TO PUBLIC
GO


CREATE TABLE dbo.Dialogu_Audit
(
        DbId INT NOT NULL,
        DialogId UNIQUEIDENTIFIER NOT NULL
)

GO
CREATE PROCEDURE dbo.usp_dergotedhena
(
        @tedhena XML
)
AS
BEGIN
        BEGIN TRY
                DECLARE @dialogu UNIQUEIDENTIFIER


                SELECT          @dialogu = DialogId
                FROM  dbo.Dialogu_Audit
                WHERE           DbId = DB_ID()
                IF @dialogu IS NOT NULL
                  AND NOT EXISTS(  SELECT          *
                                                   FROM
                                WHERE           conversation_handle = @dialogu AND state
IN ('SO', 'CO'))
                BEGIN
                DELETE
                        FROM  dbo.Dialogu_Audit
                        WHERE           DbId = DB_ID() AND DialogId = @dialogu

                        SELECT          @dialogu = NULL
                END
                IF @dialogu IS NULL
                BEGIN
                BEGIN DIALOG CONVERSATION @dialogu
                                FROM SERVICE
[tcp://10.0.0.113:4022/HR_DB/Audit/derguesi]
TO SERVICE      '//Audit/shkruaj_tedhena',                              '6A028869-F0C2-
4CE5-95AC-2652CD0B4305'
                                ON contract     [//Audit/kontrata]
                                WITH ENCRYPTION = OFF;

                        INSERT INTO dbo.Dialogu_Audit(DbId, DialogId)
                        SELECT          DB_ID(), @dialogu
                END
                ;SEND ON CONVERSATION @dialogu
                MESSAGE TYPE [//Audit/mesazhi] (@tedhena)
        END TRY
END
```

Në mënyrë që të dhënat të dërgohen asinkrone përdorim kodin e mëposhtëm i cili krijon një trigger në tabelën HR, ku të dhënat tabelare konvertohen në format XML dhe më tej nëpërmjet radhës iniciuese dërgohen ne atë destinacion për procesim te mëtejshëm

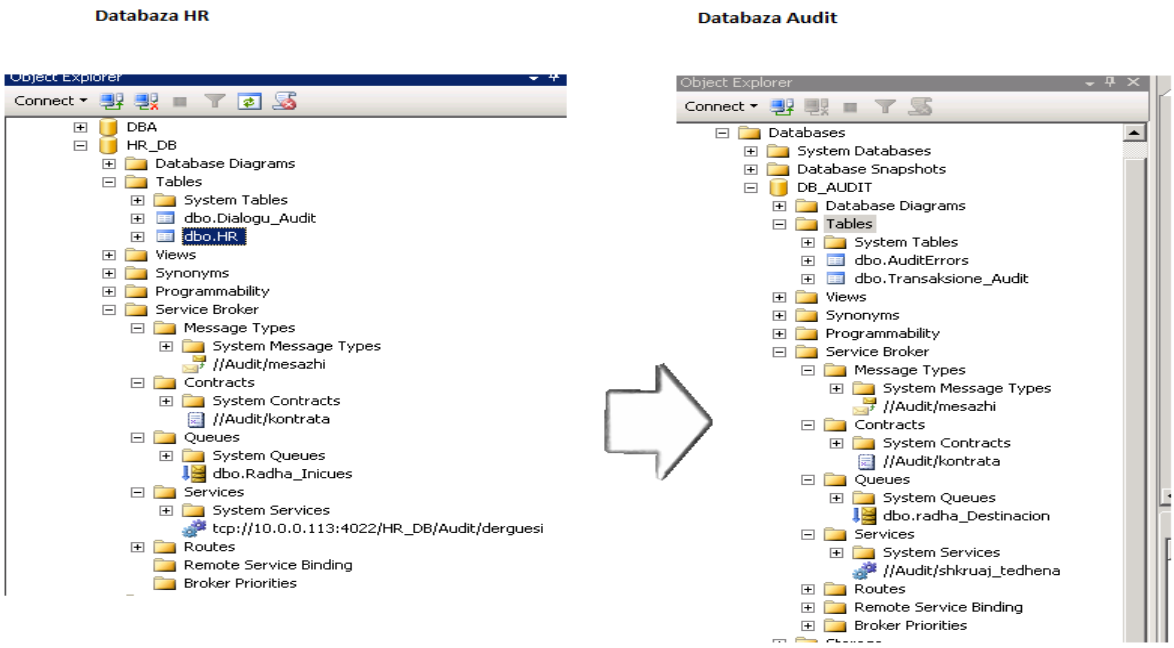
```
USE [HR_DB]
GO
/****** Object:  Trigger [dbo].[trgHRAudit]    Script Date: 01/17/2013 11:56:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[trgHRAudit]
ON [dbo].[HR]
AFTER INSERT, UPDATE, DELETE
AS
	DECLARE @permbajtja XML
	DECLARE @Veprimi VARCHAR(10)
	IF NOT EXISTS (SELECT * FROM inserted)
	BEGIN
		SELECT		@permbajtja= (select * FROM deleted AS t FOR XML
AUTO, ELEMENTS),
				@Veprimi = 'Fshij'
	END
	ELSE
	BEGIN
		SELECT		@permbajtja= (select * FROM inserted AS t FOR XML
AUTO, ELEMENTS)
		IF EXISTS (SELECT * FROM deleted)
			SELECT		@Veprimi = 'Modifikim'
		ELSE
			SELECT		@Veprimi = 'Shtim'
	END

DECLARE @TableName sysname
SELECT	@TableName = 'HR'
	SELECT @permbajtja=
		'<AuditMsg>
			<Makina_Burim>' + @@servername + '</Makina_Burim>
			<DB_Burim>' + DB_NAME() + '</DB_Burim>
			<Tabela_Burim>' + @TableName + '</Tabela_Burim>
			<UserId>' + SUSER_SNAME() + '</UserId>
			<Veprimi>' + @Veprimi + '</Veprimi>
			<TeDhenat>' + CAST(@permbajtja as NVARCHAR(MAX)) +
'</TeDhenat>
		</AuditMsg>'

	EXEC dbo.usp_dergotedhena @permbajtja
-- Dergimi i ne menyr.asinkrone , nehet mundur nga perzgjedhja e rendit te triger-ave  last
GO
EXEC sp_settriggerorder @triggername=N'[dbo].[trgHRAudit]', @order=N'Last',
@stmttype=N'DELETE'
```

```
GO
EXEC sp_settriggerorder @triggername=N'[dbo].[trgHRAudit]', @order=N'Last',
@stmttype=N'INSERT'
GO
EXEC sp_settriggerorder @triggername=N'[dbo].[trgHRAudit]', @order=N'Last',
@stmttype=N'UPDATE'
```

Paraqitja skematike midis dy bazave të dhënave do të ishte si më poshtë



Figurë 47. Konfigurimi i aplikacionit sipas mesazheve dhe radhëve

I njëjti konfigurim do të bëhet dhe në tre databazat e tjera të aplikacioneve Sms, Transferta, Asete

Bazuar në këto konfigurime shikojmë pavarësinë e aplikacionit të propozuar nga ndryshimi i llogjikës së programimit në aplikacionet e bankës dhe përfitimin në performancë, e krahasuar kjo me rastin kur nuk përdorim shërbimet e përshkruara më lart.

Pas aktiviteteve të ndryshme në aplikacione shikojmë të dhënat në databazën e qëndërzuar audit, ku nga figura më poshtë provojmë që ky sistem i qëndërzuar është implementuar i pavarur nga llogjika e programimit .

```
select * from Transaksione_Audit
```

Results | Messages

	NR	Makina_...	DB_Iniciues	Tabela	Perdorues	Veprimi	Tedhenat	Data
1	12947	SRV01	SMS_DB	SMS	jfejzaj	Shtim	<t><ID>177</ID><Klient>Klienti : 177</Klient><Telefon>069-...	2014-05-17 10:
2	13028	SRV01	SMS_DB	SMS	jfejzaj	Shtim	<t><ID>258</ID><Klient>Klienti : 258</Klient><Telefon>069-...	2014-05-17 10:
3	13109	SRV01	SMS_DB	SMS	jfejzaj	Shtim	<t><ID>339</ID><Klient>Klienti : 339</Klient><Telefon>069-...	2014-05-17 10:
4	13157	SRV01	SMS_DB	SMS	jfejzaj	Shtim	<t><ID>387</ID><Klient>Klienti : 387</Klient><Telefon>069-...	2014-05-17 10:
5	13198	SRV01	SMS_DB	SMS	jfejzaj	Shtim	<t><ID>428</ID><Klient>Klienti : 428</Klient><Telefon>069-...	2014-05-17 10:
6	13225	SRV01	SMS_DB	SMS	jfejzaj	Shtim	<t><ID>455</ID><Klient>Klienti : 455</Klient><Telefon>069-...	2014-05-17 10:
7	13252	SRV01	SMS_DB	SMS	jfejzaj	Shtim	<t><ID>482</ID><Klient>Klienti : 482</Klient><Telefon>069-...	2014-05-17 10:
8	13279	SRV01	SMS_DB	SMS	jfejzaj	Shtim	<t><ID>509</ID><Klient>Klienti : 509</Klient><Telefon>069-...	2014-05-17 10:
9	13306	SRV01	SMS_DB	SMS	jfejzaj	Shtim	<t><ID>536</ID><Klient>Klienti : 536</Klient><Telefon>069-...	2014-05-17 10:
10	13325	SRV01	HR_DB	HR	jfejzaj	Modifikim	<t><ID>2</ID><Punonjes>Punonjes --2</Punonjes><Kategori...	2014-05-17 16:
11	13326	SRV01	HR_DB	HR	jfejzaj	Modifikim	<t><ID>3</ID><Punonjes>Punonjes --3</Punonjes><Kategori...	2014-05-17 16:
12	13327	SRV01	HR_DB	HR	jfejzaj	Modifikim	<t><ID>4</ID><Punonjes>Punonjes --4</Punonjes><Kategori...	2014-05-17 16:
13	13328	SRV01	HR_DB	HR	jfejzaj	Modifikim	<t><ID>5</ID><Punonjes>Punonjes --5</Punonjes><Kategori...	2014-05-17 16:
14	13329	SRV01	HR_DB	HR	jfejzaj	Fshirje	<t><ID>5</ID><Punonjes>Punonjes --5</Punonjes><Kategori...	2014-05-17 16:
15	13321	SRV01	HR_DB	HR	jfejzaj	Shtim	<t><ID>2</ID><Punonjes>Punonjes : 2</Punonjes><Kategori...	2014-05-17 16:
16	13322	SRV01	HR_DB	HR	jfejzaj	Shtim	<t><ID>3</ID><Punonjes>Punonjes : 3</Punonjes><Kategori...	2014-05-17 16:
17	13323	SRV01	HR_DB	HR	jfejzaj	Shtim	<t><ID>4</ID><Punonjes>Punonjes : 4</Punonjes><Kategori...	2014-05-17 16:
18	13324	SRV01	HR_DB	HR	jfejzaj	Shtim	<t><ID>5</ID><Punonjes>Punonjes : 5</Punonjes><Kategori...	2014-05-17 16:
19	13029	SRV01	SMS_DB	SMS	jfejzaj	Shtim	<t><ID>259</ID><Klient>Klienti : 259</Klient><Telefon>069-...	2014-05-17 10:
20	13030	SRV01	SMS_DB	SMS	jfejzaj	Shtim	<t><ID>260</ID><Klient>Klienti : 260</Klient><Telefon>069-...	2014-05-17 10:

Figurë 48. Tabela audit pas aktiviteteve të ndryshme

Shikojmë performancën duke përdorur aplikacionin sipas mesazheve dhe radhëve e krahasuar me aplikacionet standarte të cilat modifikojnë direkt bazën e të dhënave. Për aplikacionet standarte përdorim kodin e mëposhtëm.

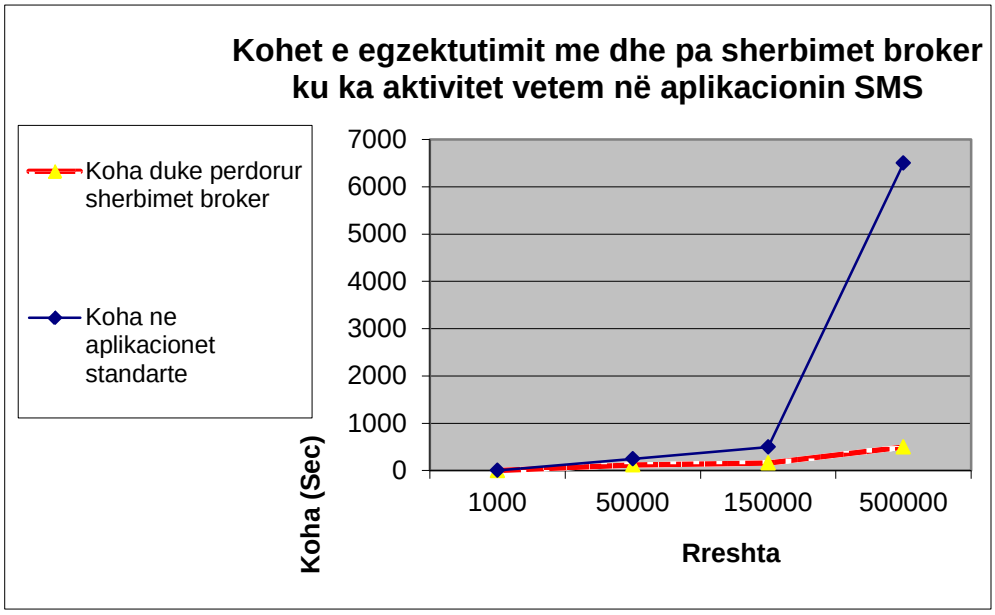
```
DECLARE @i INT
SELECT @i = 1
WHILE @i < 1000
BEGIN
       SELECT @i = @i + 1

   INSERT INTO [SRV02].DB_AUDIT2.dbo.Transaksione_Audit
 Values (@i,'SRV03','SMS_DB', 'SMS','jfejzaj','Shtim'', 'Te dhenat'
', GETDATE())
 END
```

Në tabelen më poshtë jepen kohët e egzekutimit të cilat lidhen me mbushjen e tabelës Transaksione_Audit kur ka ndërveprim vetëm në një aplikacion.

Aplikacioni	Veprime	Tipi_Veprimit	Koha me SB (sec)	Koha pa SB (sec)
SMS	1000	Shtim	2	5
SMS	50000	Shtim	120	250
SMS	150000	Shtim	160	500
SMS	500000	Shtim	500	6500

Tabelë 3. Kohët e mbushjes së logut kur ka aktivitet në një aplikacion



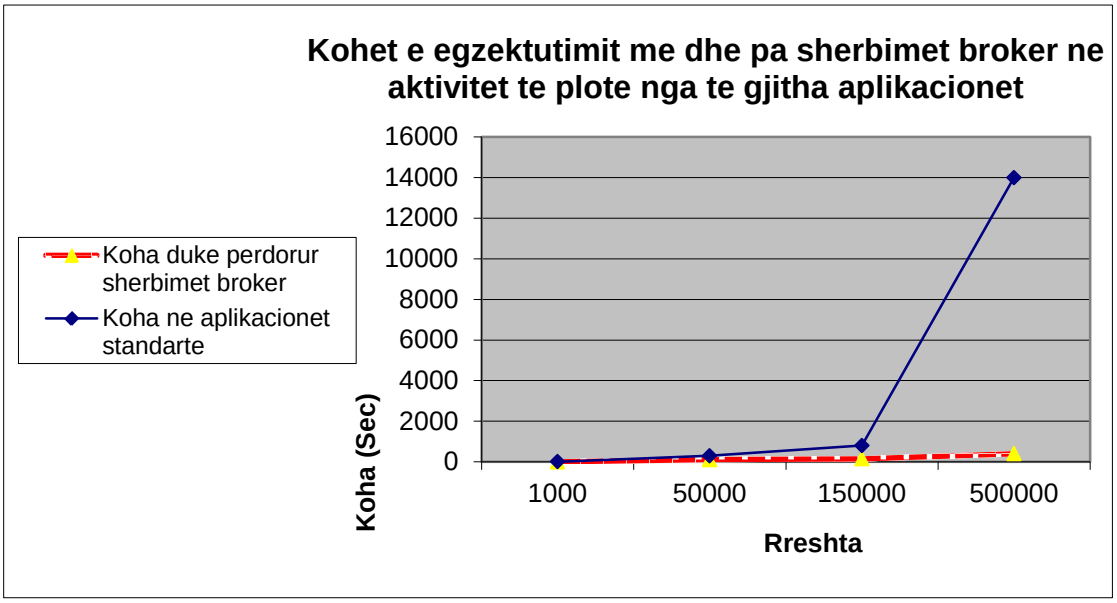
Tabelë 4. Krahasimi i performancës duke përdorur shërbimet në brendësi të databazës

Nga grafiku i performancës duket përfitimi në kohë kur krahasojmë aplikacionin e ndërtuar sipas parimit të mesazheve dhe radhëve i cili përdor një shtresë ndërmjetëse për të menaxhuar në trajtë asinkrone aktivitetin në tabelat e aplikacioneve, kundrejt aplikacioneve standarte të cilat modifikojnë direkt bazën e të dhënave.

Një krahasim tjetër është kur të gjitha aplikacionet kanë aktivitet ne të njejtën kohë dhe veprimet mund të jenë të ndërthurura (Shtim/Fshirje/Modifikim)

Aplikacioni	Veprime	Tipi_Veprimit	Koha me SB (sec)	Koha pa SB (sec)
4 aplikacione	1000	Shtim/Fshirje/Modifikim	1.5	6
4 aplikacione	50000	Shtim/Fshirje/Modifikim	110	300
4 aplikacione	150000	Shtim/Fshirje/Modifikim	150	800
4 aplikacione	500000	Shtim/Fshirje/Modifikim	400	14000

Tabelë 5. Kohët kur ka aktivitet paralel në të gjitha aplikacionet



Figurë 49. Krahasimi i performancës kur ka aktivitet paralel në të gjitha aplikacionet

Nga grafiku më lart nxjerrim si konkluzion që përfitimi në kohë kur krahasojmë aplikacionin e ndërtuar sipas parimit të mesazheve dhe radhëve rritet ndjeshëm kur në të gjitha aplikacionet ka aktivitet në të njejtën kohë dhe veprimet që inicionen nga këto mund të jenë të ndërthurura (Shtim/Fshirje/Modifikim).

Gjithashtu performaca rritet ndjeshëm kur zgjidhja e bazuar në shërbimet SB mund të kërkojë disa destinacione për të implementur strukturën load-balancing, kështu që kërkesat do të shpërndahen dhe do të kemi proçesim në paralel të mesazheve.

Kapitulli 8– Shkëmbimi i të dhënave në mënyrë standarte

8.1 Modeli Relacional dhe XML

8.1.1 Roli i XML

[34] XML (Extensible Markup Language) është një mënyrë fleksibel për të krijuar formate të perbashketa të informacionit që shpërndajnë të dy format dhe të dhënat mbi World Wide Web, intranet dhe kudo. Një mënyrë e tillë standarte e përshkrimit të te dhënave, i mundëson një përdoruesi të dërgoje një agjent inteligjent (një program) tek çdo krijues Web Site kompjuterik, te mbledhe të dhënat dhe pastaj të beje një krahasim të vlefshem. XML mund të përdoret nga çdo individ, ose një grup individesh ose kompanite që duan të shpërndajnë informacione në mënyrë të qendrueshme.

XML, një rekomandim formal nga World Wide Web Consortium (W3C), eshte e ngjashme me gjuhen e sotme te faqes Web, Hypertext Markup Language (HTML). Të dyja XML and HTML permbajnë simbole markup për të përshkruar permbajtjen e një site-i ose skedari. HTML përshkruan permbajtjen e një faqeje Web (kryesisht tekst dhe imazhe grafike) vetëm në aspektin sesi ajo duhet të shfaqet dhe ndërveproje. XML përshkruan permbajtjen në drejtim të asaj sesi të dhënat përshkruhen. Kjo do të thotë se një skedar XML mund të perpunohet thjesht si të dhëna nga një program ose mund të ruhet me të dhëna të ngjashme në një kompjuter tjeter, ose si një skedar HTML që të mund të shfaqet.

XML është me e "zgjeruar" sepse, ndryshe nga HTML, simbolet markup janë të pakufizuara dhe të vete përcaktuara.

8.1.2 XML dhe të Dhënat Relacionale

XML dhe bazat e të dhënave relacionale janë mjaft bashkepunuese në shumicen e aplikacioneve web, por në dukje këto dy modele tregohen si jobashkepunuese edhe pse shfaqen si një me vete. Modeli relacional është i bazuar në tabelat dydimensionale të cilat nuk kanë hierarki dhe rend. XML është e bazuar në pemet në të cilat renditja është mjaft e rëndësishme .
Ne një model relacional, pa hierarki dhe rend, mund të përdoret për modelim të informacionit, në XML, hierarkia dhe rendi janë mënyrat kryesore për të perfaqesuar informacionin. Edhe pse kjo është një nga dallimet me themelore midis dy modeleve, ajo nuk është aspak i vetmi.

Ne shumë mjedise, i njëjti informacion është prezent në bazën e të dhënave relacionale kur është i ruajtur ose kërkohet me anë të query-eve, por në XML vetëm kur është i shkembyer ose i shfaqur në faqet web. Këto perfaqesime jane shpesh kryesiht te ndryshme, për shkak të dallimeve ndërmjet modeleve.

Ne faqet web, XML është e përdorshme sepse struktura e XML perputhet me strukturën që përdoret për shfaqjen e informacionit në HTML. Nëse hedhim një

veshtrim në faqet web, ato shpesh përdorin një strukture hierarkike për të paraqitur të dhënat për përdoruesit –sidomos në rastet kur përdoruesit nuk duan të shikojne shumicen e tabelave dhe bëjnë join (bashkim) të kokave të tyre (head). Por shumica e këtyre të dhënave për këto faqe web vijne nga baza e të dhënave relacionale, dhe kanë nevojë të konvertohen në një hierarki të pershtashme XML.

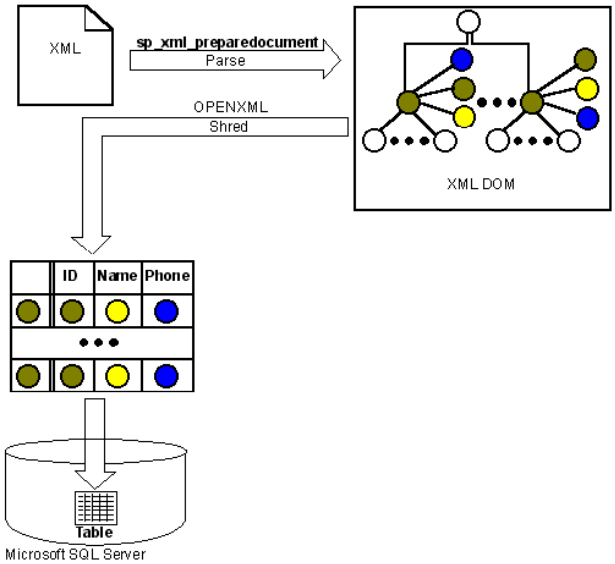
XML është zhvilluar për tu perdorur edhe si format dokument. Megjithate, karakteristikat shtese të XML, si zgjerueshmeria, mbështetja për nderkombetarizim, aftesine për të perfaqesuar të dhënat e strukturuara dhe gjysem-strukturuara, dhe lexueshmeria e lehte nga njerezit dhe makinat, i kanë dhene XML një platformë të pavarur dhe një format mjaft të njohur të perfaqesimit të te dhënave.

Përdoruesit aplikojne XML për tu gjetur zgjidhje problemeve mjaft komplekse të bizneseve, të tilla si ato që perfshijne integrimin e të dhënave. Ka shumë skenare ku është e rekomandueshme që informacionet të ruhen në formatin XML, sesa ta ruajne në tabela, dhe pastaj të kompozojne mesazhe XML nga informacioni.

Karakteristikat	Modeli i të Dhënave Relacionale	Modeli i të Dhënave XML
Te dhënat e strukturuara	Përdor tabela për ruajtjen e të dhënave në një forme kolone.Mënyrë e rekomanduar kjo për të ruajtur të dhënat e strukturuara	E dobishme kur ajo është e nevojshme për të ruajtur document ose kur skema është fleksibel ose e panjohur.
Te dhënat gjysme-te sktrukturuara	Eshte e vështirë për tu modeluar të dhënat gjysme-te strukturuar duke përdorur modelin relacional	Ofron mbështetje të shkelqyer për perfaqesimin e të dhënave gjysme-te strukturuara me variabla ose zhvillimin e skemave.
Te dhënat Marcup	Jo të përshtatshme për ruajtjen e të dhënave markup, dhe me gjere , ruatjes BLOB	E përshtatshme për magazinim e të dhënave markup si HTML,RTF dhe kshu me rradhë
Mbivendosje ose struktura e të dhënave hirarkike	Mbështet mbivendosjen e të dhënave duke përdorur tabela të shumta dhe lidhja e tyre me celesa të huaj (foreign key)por kompleksiteti i query-ve të nevojshme për kerkimin e të dhënave të mbivendosura të ruajtura në formen relacionale rritet .	Ofron mbështetje të shkelqyer për shprehjen e të dhënave si të mbivendosura ashtu edhe hierarkike
Renditja e të dhënave	Jo të ruajtura	Te ruajtura
Te dhënat Input	Homogjene	Heterogjene
Vendosja e rezultateve	Homogjene	Heterogjene

Tabelë 6. Diferenca midis modeleve të dhënave Relacionale dhe të dhënave XML

Aplikimi i XML për ruajtjen e dokumentave dhe për perfaqesimin e të dhënave gjysem të struktururar, ka cuar në zhvillimin e XML si një format i magazinimit të te dhënave që thjeshton menaxhimin e të dhënave në server.
Ky evolim, megjithate bën një problem- nxjerrjen e informacionit nga të dhënat XML të ruajtura në tabelat relacionale si BLOBs kërkohet një gjuhe query për nxjerrjen dhe formen e informacionit perfaqesuar në XML.



Figurë 50. Integrimi i XMl në SQL Server

Microsoft SQL Server 2008(2012) siguron OpenXML e cila përdoret për querying (kerkim), por është dizenjuar për harten e të dhënave XML (mapping XML) në forme relacionale, por në kete mënyrë nuk jep mbeshtjetje të plote për modelin e të dhënave XML (Figura 50)

Shumica e aplikacioneve web kanë lidhje me bazën e të dhënave dhe përdorimin e XML për të transferuar të dhënat nga baza e të dhënave në aplikacionin web dhe anasjelltas
XQuery dhe **SQL/XML**-dy standarte që përdorin mënyrën deklarative, query- por Tabelë, për kthimin e XML në të dhëna query (te kerkuara).
XQuery është XML centric, ndersa SQL/XML është SQL centrike.
SQL/XML është një zgjerim i SQL Queries dhe lejon krijimin e strukturave XML me funksione të fuqishme.(Figura 51)

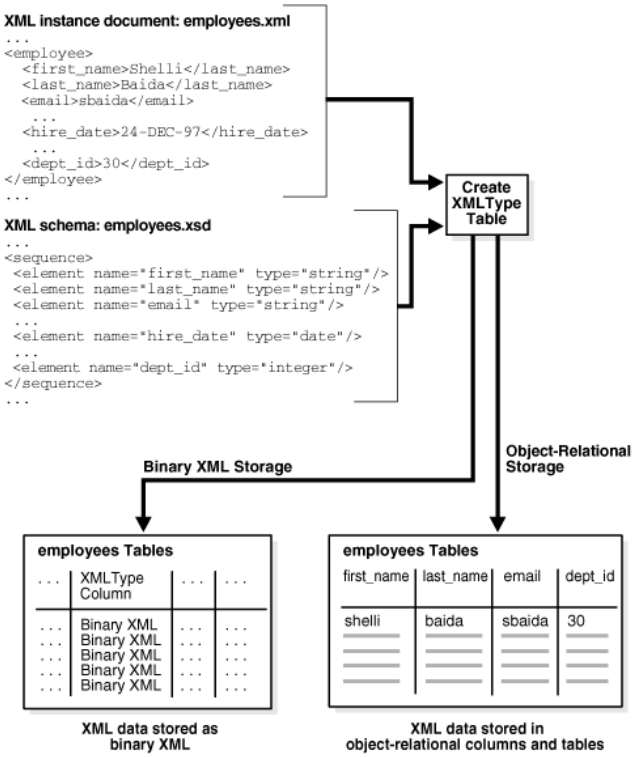
SQL/XML	XQuery

select
xmlelement(name customer,
xmlattributes(c.CustId as id),
xmlforest(c.Name as name,
c.City as city),
xmlelement(name projects,
(select xmlagg(xmlelement
(name project,
xmlattributes(p.ProjId as id),
xmlforest(p.Name as name)))
from Projects p
where p.CustId=c.CustId)))
as "customer-projects"
from Customers c
	for $c in $cust/row
	return
	<customer id="{$c/CustId}">
	<name>{ string($c/Name)}</name>
	<projects>
	{
	for $p in $proj/row
	where $p/CustId = $c/CustId
	return
	<project id="{$p/ProjId}"
	name="{$p/Name}"/>
	}
	</projects>
	</customer>

Figurë 51. Starndartet SQL/XML dhe XQuery

8.1.3 XML në Sql Server

[35]XML ka standarte të ndryshme, kështu që integrimi është bërë i vështirë nga fakti se standartet e tipit të te dhënave XML nuk janë krejtesisht të njëjta me standartet e tipit të te dhënave relacionale. XML ka atraksione të konsiderushme për DBA (ose zhvilluesi i bazës së të dhënave) për shkak se ajo siguron një mënyrë për të kaluar një shumëllojshmeri të strukturave të te dhënave si parametra, për të ruajtur ato, për query dhe për ti modifikuar. Ajo, gjithashtu, thjeshton proçesin e sigurimit të pjesës me të madhe të te dhënave feed-s. Sfida është që ta beje kete pa rritur kompleksitetin ose paqartesine e modelit të te dhënave relacionale. Atraksioni me i madh i XML nga programuesit është se ajo mund të perfaqesoje të dhëna "rowset" (tabelë e vetme) dhe të dhënat hierarkike (shumë tabela), si dhe informacion relativisht të pastrukturuara, të tilla si tekst. Kjo e bën krijimin, manipulimin dhe "vazhdueshmerine "e objekteve shumë të lehte.



Figurë 52. Një DataSet në XML

XML mund të konsiderohet një DataSet kompleks që perbehet nga disa tabela të lidhura me celesa primar dhe të huaj, në mënyrë të tillë që të mund t'i rindertoje edhe pas transmetimit. Dokumentet XML mund të perfaqesojne një ose me shumë rowsets (XML Information Set ose Infoset). Për të arritur kete, i referohet skemave XML perkates ku duhet të perfshihen në çdo dokument XML, apo fragment, të dhënat e permbajtjes (content) së XML. SQL Server ofron një skeme të magazinimit, ose library, për ruajtjen e skemave XML dhe përdor skemen e duhur për të vertetuar të dhënat e ruajtura XML si në figurën 52.

Modeli Relacional apo XML

[31]Neqoftese të dhënat janë të struktuara me një skeme të njohur, modeli relacional punon mirë për ruajtjen e të dhënave. SQL Server siguron mjetet dhe funksionalitet e nevojshme. XML është një zgjedhje e mirë neqoftese duhet një platform e pavarur për të siguruar transportin e të dhënave duke përdorur "Marcup" semantik dhe strukturor. Pervec kësaj ajo është një alternative e përshtatshme neqoftese disa nga vetite e mëposhtme merren ne konsiderate:

- Te dhënat perfaqesohen si hierarkike, në vend të referncave mes subjekteve dhe mund të jenë gjithkund rekursive.
- Rendi është i trashegueshem tek të dhënat.
- Ne mund të kërkojme të dhëna ose t'i perditesojme disa pjesë të tyre, bazuar në strukturën e tyre.

Nëse anjë nga këto kushte nuk plotesohet atehere duhet përdorimi i modelit të te dhënave relacionale. Për shembull nëse të dhënat janë në formatin XML, por kërkesa përdor bazën e të dhënave për të ruajtur apo marre të dhëna, një kolone [n]varchar(max) është e gjitha ajo çfarë na duhet. Ruajtja e të dhënave në një kolone XML ka perfitime shtese.

Arsyet për ruatjen e të dhënave XML në SQL Server

Ne vijim janë disa nga arsyet për të përdorur karakteristikat XML në SQL Server, ne vend të menaxhimit të te dhënave XML në skedaret e sistemit:

- Duam të kërkojme, modifikojme të dhëna XML në mënyrë efikase.Aksesi i të dhënave është i rëndësishem për aplikacionin. Për shembull, mund të jete e nevojshme nxjerrja e të dhënave nga disa seksione brenda një dokumenti XML, ose duam të shtojme një seksion pa ndryshuar tere dokumentin
- Kur kemi edhe të dhëna relacionale XML, ndonjehere është i nevojshem ndërveprimi ndërmjet tyre brenda aplikacionit.
- Duhet mbështetja ose suporti gjuhesor për query dhe modifikimin e të dhënave për aplikacionet cross-domain.
- Duhet server për të garantuar strukturimin e të dhënave dhe vertetimin e të dhënave sipas skemes XML
- Indeksimi i të dhënave XML për perpunimin efikas të query-ve dhe një shkallezim të mirë të tyre.
- Duhet SOAP, ADO.NET, dhe OLE DB për aksesin e të dhënave XML.

- Përdorimi i funksioneve administrative të serverit të bazës së të dhënave për menaxhimin e të dhënave XML si per shembull, back-up,recovery dhe replikim.
- Nëse asnjë nga këto kushte nuk plotesohen ose përshtaten , atehere është mirë për ruajtjen e të dhënave jo XML në objekte të medha të tipit si: [n]varchar(max) or varbinary(max).

8.2 Kalimi nga te Dhënat Tabelare në XML

Table 2 XML

Si të konvertojme të dhënat tabelare në format XML

[32] SQL Server, duke filluar nga versioni SQL 2005 ka shtuar një tip të ri të dhënash për XML.

Kjo ka bërë të mundur që të dhënat XML të ruhen në vlerat e një kolone të tabelës, ku mund të aplikohen gjithashtu rregulla për validimin e të dhënave që mund të futen në një dokument XML. Kjo realizohet nëpërmjet skemave XML. Gjithashtu mund të deklarojme variabla të tipit XML dhe ti përdorim ato në skriptet XML.

Për shembull kodi i mëposhtëm ku deklarohet një dokument XML :

```
DECLARE @doc xml
SET @doc = '<?xml version="1.0" ?>
 <Autori AutoriId="1000" Emri="Ndre" Mbiemri="Mjeda">
  <Librat>
   <Libri LibriId="Z1234567" />
    <Libri LibriId="X1234567" />
     <Libri LibriId="Y1234567" />
      <Libri LibriId="W1234567" />
  </Librat>
</Autori>'

SELECT @doc
```

Ne formatin XML jep këto rezultate:

```
<?xml version="1.0" ?>
 <Autori AutoriId="1000" Emri="Ndre" Mbiemri="Mjeda">
  <Librat>
   <Libri LibriId="Z1234567" />
    <Libri LibriId="X1234567" />
     <Libri LibriId="Y1234567" />
      <Libri LibriId="W1234567" />
  </Librat>
</Autori>
```

Ne permbajtjen e mësipërme XML, **Autori** edhe një element XML, (nganjehere referohen edhe si **nyje** "node" XML). Ndersa AutoriId, Emri dhe Mbiemri janë **atributet e elementit Autori**.
Elementi Autori permban elementin **Librat**.
Elementi Librat permban elemente **Libri**. Elementet Libri kanë atributin LibriId.
Ky dallim midis **elementeve** dhe **atributeve** është i rëndësishem, por kete do ta shohim kur të përdorim FOR XML PATH, në seksionin me poshtë.

Do të shikojme së pari, si ti konvertojme të dhënat tabelare në format XML. Duke patur parasysh nëse duhet të konvertojme listen e punonjesve nga tabela Punonjesit në një format XML apo nëse duhet të eskportojme porosite (Porosite) në XML për ti shkembyer me një tjeter aplikacion, për shembull me aplikacioni e finances për ti ndjekur me tej porosite për efekte pagese, kontabiliteti etj.

Për të konvertuar të dhënat tabelare në XML, ekziston një klauzole e komandes SELECT, që është FOR XML.
Klauzola FOR XML ka disa mënyra përdorimi dhe secila prej këtyre ka disa opsione, si për shembull opsioni për ti shtuar një element rrenjë (root) dokumentit XML.
Mënyrat e përdorimit të FOR XML janë kater:

- FOR XML RAW
- FOR XML AUTO
- FOR XML PATH
- FOR XML EXPLICIT

Për me tepër është e mundur që komandat që kthejne XML të integruar brenda komandave (query) që kthejne XML ose të dhëna tabelare, por këto do ti kemi serish në shqyrtim pasi të shpjegojme mënyrat e përdorimit të FOR XML.

FOR XML RAW

Tipi RAW transformon cdo rresht në rezultatin e kërkeses nga baza e te dhënave në një element XML që ka identifikuesin gjenerik <**row**>, ose elementin e ofruar në mënyre opsionale **name**. Si default, cdo vlere e kolones në rowset që nuk jane null, lidhet me një atribut te elementit <**row**>. Nëse direktiva **ELEMENTS** I shtohet pjesës FOR XML, cdo vlere e kolones lidhet me një nënelement te elementit <**row**>.

Përdorimi i FOR XML RAW

Sintaksa e përdorimit të FOR XML është shumë e thjeshte dhe memorizohet lehte.
Ne fund të query-t vendoset fjala FOR XML dhe metoda e përdorur.

Një query e zakonshme që afishon të dhënat, nëse ka në fund klauzolen FOR XML , e prodhon rezultatin në format XML.

FOR XML RAW është mënyra me e thjeshte e prodhimit të rezultateve në format XML. Për ta përmirësuar rezultatin mund të përdorim opsionet e FOR XML RAW

FOR XML RAW, Elements

Opsioni **ELEMENTS** i afishon kollonat e tabelës si elemente në vend të atributeve.

Ky është një rast kur FOR XML RAW, paraqitet kufizues. Në kete situate kemi dy zgjidhje:

1. Te integrojme komandat SQL që prodhojne XML brenda njëra tjetrës duke përdorur opsionin **TYPE**
2. Te përdorim **FOR XML AUTO -** që është me i pershtashem pikerisht për ato raste kur kemi një rezultat të një query me JOIN

FOR XML AUTO

Ne mënyrë të veçantë nëse rezultati i kërkesave vjen nga një JOIN mbi dy ose me shumë tabela, FOR XML AUTO është më efektiv, pasi ai nuk përsërit të dhënat e tabelës prind.

FOR XML PATH

Një tjeter mënyrë për të konvertuar të dhënat tabelare në XML është FOR XML PATH. Kjo mënyrë i eksporton të dhënat në XML në një strukturë të ngjashme me direktorite në sistemin operativ.

/ (slash) përdoret për të ndare nyjet nga njëra tjetra
. Simbolizon nyjen korrente
.. Simbolizon nyjen prind të nyjes korrente

Nëpërmjet FOR XML PATH mund të krijojme edhe nyje që nuk kanë korrespondence direkte me kollonat e tabelave. Kjo është ndonjëherë e nevojshme për të ndërtuar XML sipas një skeme të përcaktuar paraprakisht nga partneri me të cilin duam të shkëmbejme të dhëna në formatin XML.

Po nëse kemi të dhëna nga disa tabela?
Ne kete rast do të ishte e lehte të kombinonim një PATH XML me një RAW ose me një tjeter PATH XML duke nderfutur komandat (query) SQL nëpërmjet opsionit TYPE.

FOR XML EXPLICIT

Konvertimi i të dhënave tabelare në XML nëpërmjet FOR XML EXPLICIT

Konvertimi i të dhënave në XML nëpërmjet FOR XML EXPLICIT, është një proçes paksa me i komplikuar, por na jep kontroll të plote mbi mënyrën se si do të konvertohen të dhënat XML. në shumicen e rasteve na mjaftojne tre mënyrat e ilustruara deri me tani RWA, PATH dhe AUTO.

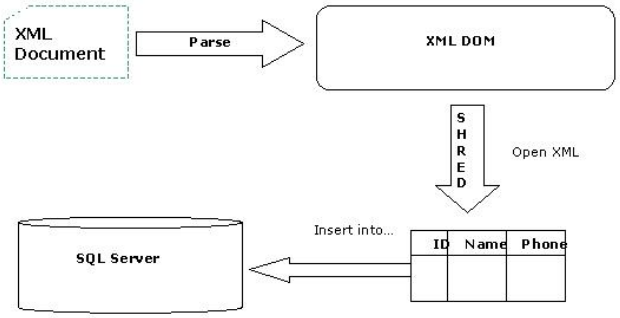
Konvertimi nëpërmjet mënyrës EXPLICIT, bazohet në tabelat universale. Tabelat universale kanë dy kollona:
TAG dhe PARENT. Kollona TAG ka vleren e nivelit që kanë rreshtat e tabelës dhe kollona PARENT ka vleren NULL për nivelin e pare të pemes XML dhe vleren e kollones TAG të rreshtave që sherbëjnë si Prind.

8.3 Kalimi nga XML në te Dhënat Tabelare

Open XML

Si të lexojme të dhënat XML dhe ti paraqesim ato në forme tabelare - OPEN XML

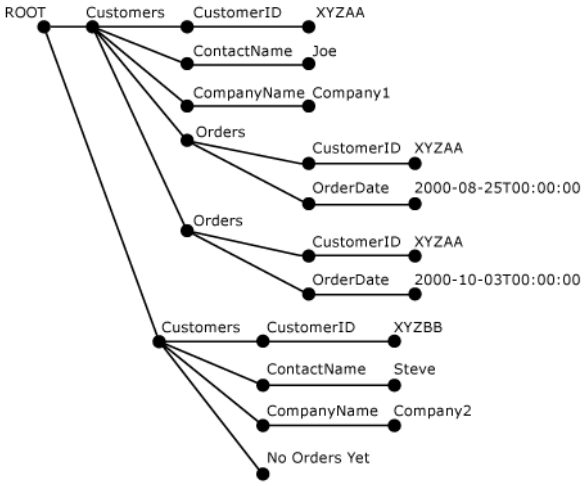
[36]OPENXML ofron një mënyrë e thjeshtë për të përdorur një dokument XML si një burim te dhënash për proçedurat . OPENXML lejon të dhënat në XML dokument që do të trajtohen vetëm si kolonat dhe rreshtat e tabelës së bazës së të dhënave dmth, taget XML si kolona dhe vlerës si rreshtave (Figura 53)



Figurë 53. OPEN XML

Të dhënat mund të futen / përditësohen shumë shpejt dhe me efikasitet, pa hapa të shumta në bazën e të dhënave.
[33] Në kete pjesë do të shikojme konvertimin mbrapsht, nga të dhënat XML në të dhëna tabelare. Për kete ekziston një komande e SQL OPENXML, e cila merr si argument një dokument XML dhe pozicionohet në të duke lexuar të dhënat sipas PATH XML.

Perpara se të mund të lexojme brenda një variabli XML me OPENXML të ndertojme strukturën e pemes së dokumentit XML në memorje.
Për kete SQL ka një procedure të sistemit **sp_xml_preparedocument.**



Figurë 54. Proçedura SP XML Prepare Dokument

Sp_xml_preparedocument lexon tekstin XML që merr si input, e lexon ate nëpërmjet lexuesit MSXML (Msxmlsql.dll), dhe kthen në një variabel të tipit INT, një shenjues mbi strukturën e dokumentit XML. Dokumenti i kthyer është një paraqitje në forme peme e nyjeve të dokumentit XML: elementet, atributet, etj.

Kapitulli 9–SOA dhe bazat e të dhënave me kapacitet të lartë (Big Data)

9.1 Bazat e të dhënave me kapacitet të lartë

Ndërmarrjet moderne përdorin baza të dhënash me misionit kritik që përmbajnë disa qindra gigabajt, dhe shpesh disa terrabajt të dhënash. Këto ndërmarrje sfidohen nga mbështetja dhe mirëmbajtja e kërkesave të bazave të të dhënave me kapacitet shumë të lartë (VLDB - Very Large Databases), dhe duhet të hartojnë metoda për të përmbushur këto sfida.

Çfarë është një bazë të dhënash me kapacitet të lartë?
[37] Përcaktimi i bazës së të dhënave me kapacitet të lartë është gjithmonë në ndryshim, përderisa IT mbledh më shumë të dhëna dhe meqenëse hardware dhe software evoluojnë për të mbajtur sasi më të mëdha të të dhënave.
Termi bazë të dhënave me kapacitet të lartë mund të klasifikohet nga një shumëllojshmëri kriteresh, siç është përcaktuar në tabelën më poshtë:

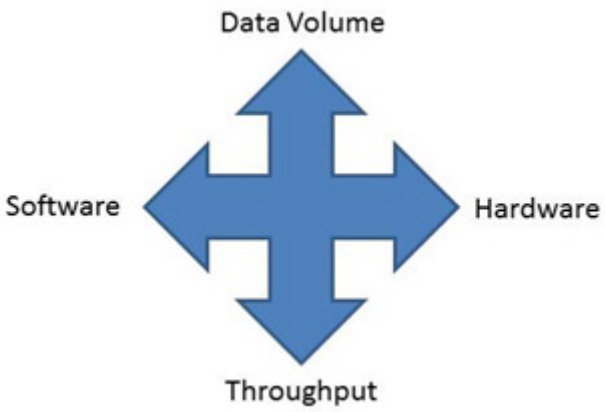
Bazat e të dhënave me kapacitet të ulët	Bazat e të dhënave me kapacitet mesatar	Bazat e të dhënave me kapacitet të lartë
Përshtatet në memorje	Përshtatet në një server të vetëm	Përhapet mbi server të shumtë
Nuk kërkohet DBA	1 DBA	Disa DBA
$<10^5$ Rekorde	10^5– 10^7 Rekorde	$>10^7$ Rekorde
<10 GB e të dhënave	10GB – 40GB e të dhënave	>40GB e të dhënave
Nuk ka particionime	Particionime minimale	Ndarje masive
-- Performanca nuk është problem --		Performanca është problem!

Tabelë 7. Krahasime midis bazave të dhënave me kapacitet të lartë

Nëse do ta bëhej update kjo listë në 3-5 vjet, të gjitha matjet fikse ndoshta do të rriteshin nga një faktor 10X deri 100X, apo edhe më shumë. Ndoshta përshkrimi më i përshtatshëm për një "bazë të dhënash me kapacitet të lartë" është ai që na shkakton mundim duke na detyruar që të marim në konsideratë dhe opsione të tjera.

Faktorët që përcaktojnë një bazë të dhënash me kapacitet të lartë

Termi "bazë të dhënash me kapacitet të lartë" është një objekt në lëvizje për shkak se teknologjia vazhdon të zhvillohet. Ajo që konsiderohet me kapacitet të lartë kur ekzekutohet nga një hard disk mund të duket e menaxhueshme kur ekzekutohet në një Solid State Disk (SSD), kur ekzekutohet në memorje, apo kur ekzekutohet në një bazë të dhënash elastike NoSQL ose ScaleDB për MySQL. Si rezultat, mund ta konsiderojmë madhësinë e bazës së të dhënave të jetë bashkëveprim i katër faktorëve të më poshtëm:



Figurë 55. Faktorët që ndikojnë në bazat e të dhënave me kapacitet të lartë

Volumi i të dhënave (Data Volume): Sasia e të dhënave të përcaktuara nga numri i rekordeve, tabelave, terrabajtëve/pedabajtëve, etj.

Hardware: Ekzekutimi i një baze të dhënash me kapacitet të ulët mbi një server jashëzakonisht të kufizuar do të dukej si një bazë të dhënash (problematike) me kapacitet të lartë.

Xhiroja (Throughput): Ky është zhargoni i bazës së të dhënave për përdorim. Nëse kemi një bazë të dhënash me kapacitet të ulët por i shërbehet 10 milion përdoruesave aktualë, do të duket si një bazë të dhënash me kapacitet të lartë. Matjet e niveleve të përdorimit janë konsideruar si xhiro.

Software: Ky mund të konsiderohet duke përfshirë së bashku sistemin e menaxhimit të bazës së të dhënave (DBMS - database management system) që po përdorim, si edhe implementimin e bazës së të dhënave në vetvete. Ky implementim mund të marrë përdorim të gjerë të I/O, rrejtit, apo proçeseve intensive të CPU-së. Mund të varet gjithashtu edhe nga përdorimi jonë i optimizimeve siç janë përshembull indekset.

Baza jonë e të dhënave është aq e mirë sa edhe dobësia e këtyre katër faktorëve, por mund ti kompensojmë për dobësitë. Për shembull, nëse hardware-i jonë ka një disk me kapacitet të ulët mund të përdorim kompresimin. Ose nëse kemi pak kapacitet në RAM, në vend të një DBMS-je në memorje, do të donim një që është më shumë efikas në RAM.

9.2 Problematikat në Bazat e të dhënave me kapacitet të Lartë (VLDB)

9.2.1 Koncepti i Particionimit

Particionimi trajton çështjet kryesore në mbështetjen e indekseve dhe tabelave shumë të mëdha duke i dekompozuar ato në pjesë më të vogla dhe më të menaxhueshme të quajtura ndarje, të cilat janë tërësisht transparente për një aplikim. Përcaktimet SQL query dhe Data Manipulation Language (DML) nuk duhet të modifikohen për të aksesuar tabelat e ndara. Megjithatë, pasi particionimet janë vendosur, përcaktimet e

Data Definition Language (DDL) mund të aksesojnë dhe të manipulojnë particionime individuale në vënd të të gjithë tabelave apo indekseve. Kjo tregon se si particionimi mund të lehtësojë menaxhueshmërinë e objekteve të mëdha të bazës së të dhënave.

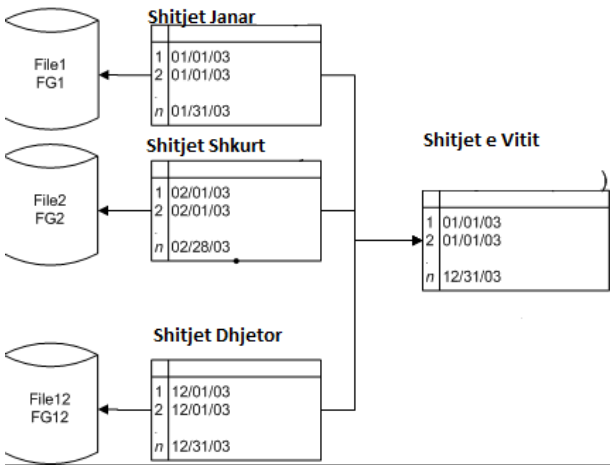
Çdo particionim i një tabelë apo indeksi duhet të ketë të njëjtat atribute logjike, të tilla si emrat e kolonës, tipet e të dhënave dhe përmbajtjet, por çdo particionim mund të ketë atribute fizike të ndara, të tilla si aktivizimi ose çaktivizimi i kompresimit, atribute të ruajtes fizike, dhe hapësira tabelash.

Particionimi është i nevojshëm për shumë lloje të ndryshme të aplikacioneve, veçanërisht për aplikacionet që menaxhojnë volume të mëdha të të dhënave. Sistemet OLTP shpesh përfitojnë nga përmirësimet në përshtatshmëri dhe disponueshmëri, ndërsa sistemet data warehouse përfitojnë nga performanca dhe menaxhueshmëria.

Particionimi ofron këto avantazhe:

- Mundëson operacione të menaxhimit të të dhënave të tilla si ngarkesa të të dhënave, krijimin dhe rindërtimin e indekseve, si edhe backup dhe rikuperim në nivel particionimi. Kjo rezulton në mënyrë të konsiderueshme në reduktimin e kohës për këto operacione.
- Përmirëson performancën e query-t. Shpesh rezultatet e një query mund të arrihen më mirë duke aksesuar një nënbashkësi të particionimit, sesa të gjithë tabelën. Për disa query, kjo teknikë (e quajtur shkurtim i particionimit) mund të sigurojë fitime në performancë.
- Në mënyrë të konsiderueshme zvogëlon ndikimin e kohës joproduktive të planifikuar për operacionet e mirëmbajtjes.
  Pavarësia e ndarjeve për operacionet e mirëmbajtjes së particionimit na lejon të kryejmë operacione të njëkohshme të mirëmbajtjes në particionime të ndryshme të së njëjtës tabelë apo indeksi. Mund të ekzekutojmë gjithashtu operacione të njëkohshme SELECT dhe DML përkundrejt particionimeve që janë të paprekura nga operacionet e mirëmbajtjes.
- Rrit disponueshmërinë e bazave të të dhënave me misionit kritik në qoftë se tabela dhe indekse kritike janë të ndarë në particionime për të reduktuar dritaret e mirëmbajtjes, orët e rikuperimit, dhe ndikimin e dështimeve.
- Ekzekutimi paralel ofron avantazhe të veçanta për të optimizuar shfrytëzimin e burimeve, dhe për të minimizuar kohën e ekzekutimit. Ekzekutimi paralel ndryshe nga objektet e particionuara është i rëndësishëm për shkallëzueshmërinë në një mjedis të klasterizuar. Ekzekutimi paralel mbështetet për query si edhe për DML dhe DDL.

Particionimi mundëson akses të shpejtë të të dhënave. Nëse një bazë të dhënash ka 10 GB ose 10 TB të dhëna, particionimi mund të përmirësojë aksesin e të dhënave lidhur me rritjen e madhësisë së të dhënave. Particionimi mund të implementohet pa kërkuar modifikime për kërkesat. Për shembull, mund të konvertojmë një tabelë të pa particionuar në një tabelë të particionuar pa pasur nevojë për të ndryshuar ndonjë nga përcaktimet SELECT apo DML që aksesojnë atë tabelë. Nuk kemi nevojë ta rishkruajmë kodin e aplikacionit për të përfituar nga particionimi. Në figurën 56, është paraqitur particionimi i tabelës shitje sipas muajve të vitit.



Figurë 56. Particionimi i tabelës Shitje per ç'do muaj te vitit

9.2.2 Përfitimi në Performancë me anë të Particionimit

Rritja e madhësisë së bazave të të dhënave mund të përballet me administratorët e bazës së të dhënave me çështje të cilat lidhen me performancën.
Shumica e tregtarëve që shesin software për produkte DBMS kanë përfshirë mjete dhe teknika që e ndihmojnë administratorin e bazës së të dhënave të përmirësojë performancën në bazën e të dhënave. Në artikullin [38] është eksperimentuar një nga teknikat e përdorura për të përmirësuar performancën e bazës së të dhënave, e quajtur "particionimi i tabelës". Kjo teknikë do të shërbejë në punimin tonë për të ndarë volumin e madh të të dhënave që krijohen nga bazat e të dhënave të shpërndara në rrjet, për rritjen e performancës dhe për të menaxhuar sa më mirë arshivën e të dhënave.
Testi do të bëhet në ambjentin SQL Server, i cili është një nga sistemet më të përdorura të menaxhimit të e bazës së të dhënave. Janë përshkruar hapat për të implementuar e particionimit të tabelës në SQL Server 2008 R2. Një tabelë e ndarë me dy particionime do të krijohet për të testuar performancën e çdo query në çdo particionim.
Një proçes i popullimit të të dhënave do të aplikohet në tabelë në mënyrë që ti plotësojë particionimet me sasi të diferencuara të të dhënave. Particionimi më i madh do të quhet "arkiv" ndërsa particionimi më i vogël do të quhet "aktual".
Do të krijohet një tabelë krahasimi e cila ruan sasinë e kohës që kërkohet për të ekzekutuar secilin prej query-ve. Gjashtë teste për çdo query do të ekzekutohen dhe do të përdoret mesatarja e tyre në mënyrë që të sigurojnë rezultate të sakta.

Mjetet e përdoruara
Mjedisi Hardware në te cilin u kryen testet është me specifikimet si më poshtë:

Server HP DL360 G5 Server
Modeli : DL3605
Shpejtesia CPU : 3 GHz
Memorje: 2 GB Number
Numri i CPU : 1
Sistemi Operimit: Windows Server 2003 32-Bit

Aplikacionet e përdorua në Server:

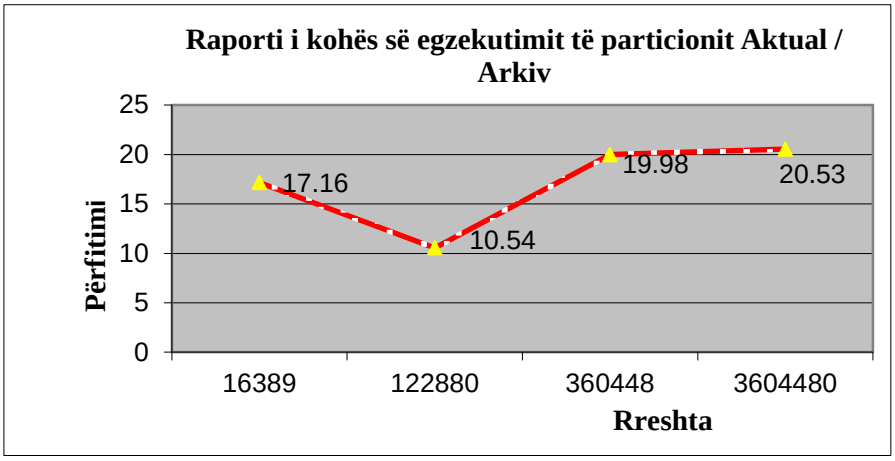
Sistemi i menaxhimit i bazës së te dhënave është perdorur version SQL Server 2008 R2 Enterprise
Si editor per kërkesat është perdorur SQL Server 2008 Management Studio.
Si mjet per te matur kohet e egzekutimit kemi perdorur SQL Server Profiler.

Metodologjia

Tabela transaksione_audit ne paragrafin 7.3, do të shërbejë për ndarjen në dy particione arkiv dhe aktual. Ne bazë të vlerave të fushës çelës (fusha datë) particionet do të popullohen me të dhëna perkatesisht si në tabelën më poshtë, ku tendenca e volumit të të dhënave në arkivë shkon mbi 95%.
Testet do te kryhen në kater faza te popullimi te tabelës dhe do te llogaritet perfitimi i cili llogaritet si raport i kohës së egzekutimit në particionin Arkive ndaj kohës së egzekutimit në particionin Aktual

Particioni/Testi	Nr_rreshta	Koha Mesatare e egzekutimit te kërkesës (ms)	Permirësimi
Arkive Testi1	16385	1018.17	
Aktual Testi1	4	59.33	17.16
Arkive Testi2	114688	6407	
Aktual Testi2	8192	607.83	10.54
Arkive Testi3	344064	19547.5	
Aktual Testi3	16384	978.17	19.98
Arkive Testi4	3440640	205003.17	
Aktual Testi4	163840	9984.5	20.53

Tabelë 8. Koha mesatare e egzekutimit te kërkeses për ç'do particion



Figurë 57. Përfitimi në Performancë duke particionuar të dhënat

Nga grafiku 57, si konkluzion mund të nxjerrim që perfitimi që arrihet duke particionuar te dhënat ka tendecen te rritet linearisht me sasine e te dhënave me te cilat popullohet tabela, pra sa me te medha te jenë tabelat në një baze te dhënash aq me shumë mund te perfitojme nga particionimi.
Particionimi i tabelës është një teknikë e cila permirëson ndjeshëm performancën e aksesit te tabelave te medha. Një nga avantazhet e medha te kesaj teknikë është fakti që te dhënat e tabelës jane llogjikisht brenda së njëjtës tabele, ndërkohë që fizikisht mund te jenë te ndara në disqe te ndryshme, gje e cila rrit performancën dhe me e rëndësishmja nuk kerkon ndryshime në aplikacione te cilat komunikojne me bazën e te dhënave.

9.2.3 Shërbimet Data Mining në të dhënat me kapacitet të lartë

Te dhënat Mining eshte nje proces i rendesishem per te nxjerre informacion dhe njohuri nga te dhënat. Informacioni dhe njohuria e nxjerre mund te perdoren per te ndertuar startegjite e kompanive, duke mbeshtetur proceset kritike te tilla si analiza e marketingut, tendencat e sjelljes te konsumatoreve, analiza llogaria ekonomike, per zbulimin e mashtrimeve, dhe analiza e rrjetit. Me qellim per te rritur efektivitetitn e ketyre startegjive, te dhënat mining fokusohen ne zbulimin e modeleve te fshehura tek te dhënat.
Volumi dhe dimensioni i te dhënave bejne te mundur analizen e te dhënave ne sherbim te biznesit. Kjo eshte arsyeja pse metodologjite e bazuara ne kompjuter jane aplikuar per nxjerrjen e informacionit te vlefshem .
Inteligjenca e bizneseve mbeshtetet ne te dhënat mining. Te dhënat mining eshte procesi i analizimit te te dhënave per te gjetur modele te fshehura apo lidhje qe nuk mund te zbatohen ne cast. Mund te jete ose nje process manual ose mund te perdoret per kërkesat ad-hoc apo aplikacioneve te sofistikuar te cilet zbatojne algoritme te ndryshme. Keto algoritme gjejne modele, klasifikojne te dhënat aktuale ose parashikojne sjelljet e ardhshme.

Të dhënat Mining

[44]Data mining do te thote qe te nxjerrin ose te nxjerre njohuri nga sasi shumë të madhe te te dhënave qe jane te ruajtura ne magazinat (repository) e llojeve te ndryshme te tilla si bazat e te dhënave relacionale, te dhënat warehouse, baza e te dhënave te transaksioneve, skedare te ruajtur (flat file) dhe World Wide Web.
Baza e te dhënave te transaksionit perbehet nga nje skedar ku cdo rresht qendron per nje transaksion. Zakonisht keto baza te dhënash nuk permbajne struktura te mbivendosura relacionale, per kete arsye jane te ruajtur me shpesh ne nje skedar. Cdo transaksion eshte identifikuar nga nje identifikues unik. Te dhënat warehouse perdoren per te ruajtur te dhëna te integruar qe pershkruajne subjekte te ndryshme. Te dhënat mund te vijne nga burime te shumta (baze te dhënash, skedare) keshtu qe ata

jane pastruar (cleaned) se pari dhe pastaj jane transformuar, integruar dhe te ngarkuar ne destinacionin e fundit. Te dhënat warehouse ruajne te dhëna historike

Detyrat paresore të të dhënave mining janë:

1. **Klasifikimi** - te dhënat jane klasifikuar ne nje ose me shume klasa nepermjet nje funksioni parashikues te lexuarit
2. **Regresioni** - zbulimi i nje funksion parashikues te mesuarit, e cila lidh nje object te te dhënave me nje variabel me vlere te vertete
3. **Clustering** - nje detyre e zakonshme pershkruese ne te cilen kerkon te identifikoje nje grup i caktuar i kategorive apo te grupimeve per te pershkruar te dhënat .
4. **Permbledhja** - nje grup ose nengrup te dhënash qe merr te dhëna te pershkruara gjetur gjatë procesit mining
5. **Varesi Modelimi** - gjetjen e nje modeli lokal qe pershkruan varesi te rendesishme mes variablave apo mes te vlerave te nje funksioni ne nje grup te dhënash ose ne nje pjese te nje seri te te dhënave.
6. **Ndryshimi dhe Shmangia e Zbulimit** - Zbulimi i ndryshimeve me te rendesishme ne bashkesine e te dhënave

Modelet e te dhënave mining duhet te ndihmojne ne marrjen e vendimeve. Per kete arsye , ata duhet te jene ne ate menyre qe njerezit ti besojne vendimet e tyre mbi to . Qellimet e saktesise se modelit dhe saktesise se interpretimit te saj jane disi kontardiktore. Zakonisht modelet e thjeshta jane me te interpretueshme, por ata jane me pak te sakta. Metodat modern te te dhënave mining, priren te japin rezultate shume me te sakte duke perdorur model te larte dimensional. Problemi i interpretimit te ketyre modeleve, gjithashtu shume e rendesishme, eshte konsideruar si nje detyre e veçantë, me teknika te vecanta per te vertetuar rezultatet. Perdoruesit nuk do te mund te gjejne qindra faqe te dobishme te rezultateve numerike. Ata nuk do te jene ne gjendje per te kuptuar, permbledhin, interpretojne dhe perdorimin e tyre per marrjen e vendimeve te sukseshmes.

9.2.4 SOA dhe të dhënat me kapacitet të lartë

Të dhënat me kapacitet të lartë ruhen në bazat e të dhënave NoSQL. NoSQL është zakonisht jo-relacionale, e shpërndarë, ‘open source’, e shkallëzuar horizontalisht, skemë e lirë, e qëndrueshme, dhe masive në aspektin e volumit të të dhënave, shpejtësisë, dhe shumëllojshmërive.

Të dhënat me kapacitet të lartë janë duke e realizuar premtimin e tyre. Koha e përpunimit të ngarkimit tani mund të reduktohet duke ngarkuar të dhëna të papërpunuara në Hadoop. Kompanitë e sigurimeve mbledhin shpërblimet duke përdorur të dhënat e grumbulluara nga sensorë që regjistrojnë modelet lëvizëse dhe kilometrazhin.

[40] Volumi i të dhënave po rritet në mënyrë eksponenciale. Në vitin 2010, volumi i të dhënave në botë kaloi një zetabajt (njëshi i ndjekur nga njëzet e një zero), ose një miliard terabajt. Deri në vitin 2015, volumi global llogaritet të jetë rreth tetë zetabajt. Një miliard objekte të reja Facebook-u dhe 200 milionë tweets çdo ditë sugjerojnë së bashku edhe volumin e plotë edhe shumëllojshmërinë e të dhënave me kapacitet të lartë.

9.3 Arkitektura "Të dhënat me kapacitet të lartë-si-shërbime" Big Data as-a-Service [BDaaS]

Le të marrim në konsideratë se çfarë do të thotë arkitekt SOA-bazuar në të dhënat me kapacitet të lartë. Së pari duhet të kujtojmë se SOA është më tepër një set kufizimesh dhe parimesh projektimi, sesa vetëm një grup i shërbimeve Web.[41] Komponentizimi, ndarja e problemeve, thjeshtësia, dhe shtresëzimi janë parime të tjera që mbështesin SOA dhe të dhënat me kapacitet të lartë. Ndërsa software-i i të dhënave me kapacitet të lartë vazhdon të përmirësohet, ndryshimet mund të bëhen në ndërfaqen e përdoruesit, në komunikimet, në ruajtjen e të dhënave, dhe në shtresat e përpunimit të detyrës, pa pasur nevojë për ta rindërtuar të gjithë arkitekturën.

Inteligjenca e biznesit (BI) ndan me të dhënat me kapacitet të lartë shumë prej zgjidhjeve të problemeve që kanë lejuar BI të bëhet shërbim i orientuar, i cili teknikisht përfshin përdorimin e përshtatësave, metadata-ve, data mart-eve, dhe pjesën tjetër të mbetur. Konceptualisht, kjo do të thotë të ndjekësh shkathtësinë duke punësuar të njëjtat lloje të prezantimeve, proçeseve të biznesit, dhe shtresave të shërbimeve të të dhënave. Sfida për integrimin e të dhënave me kapacitet të lartë në një organizatë data warehouse është më tepër një çështje e menaxhimit të ndryshimit organizativ sesa një problem teknik.

Përpunimi i ngjarjeve komplekse (CEP- Complex event proçessing) përfshin teknologjitë që kanë kapacitetin për të analizuar të dhënat në lëvizje në përgjigje të ngjarjes së trigerave ose ndryshime në modelet e të dhënave .

Parimet SOA përfshijnë bashkim të lirshëm të shërbimeve, lidhje reciproke një-për-një midis ofruesit dhe konsumatorëve, dhe zakonisht sjellje sinkrone (kërkesë-përgjigje). Në të kundërt, arkitektura e ngjarjes së shtyrë karakterizohet nga ndërveprime të tërhequra, komunikime shumë-me-shumë, veprime ngjarjesh të kontrolluara, dhe operacione asinkrone. BDaaS shfrytëzon parimet që janë marë nga SOA dhe EDA. BDaaS mund të konsiderohet si një SOA asinkron, me një marrëdhënie plotësuese midis shërbimeve dhe ngjarjeve.

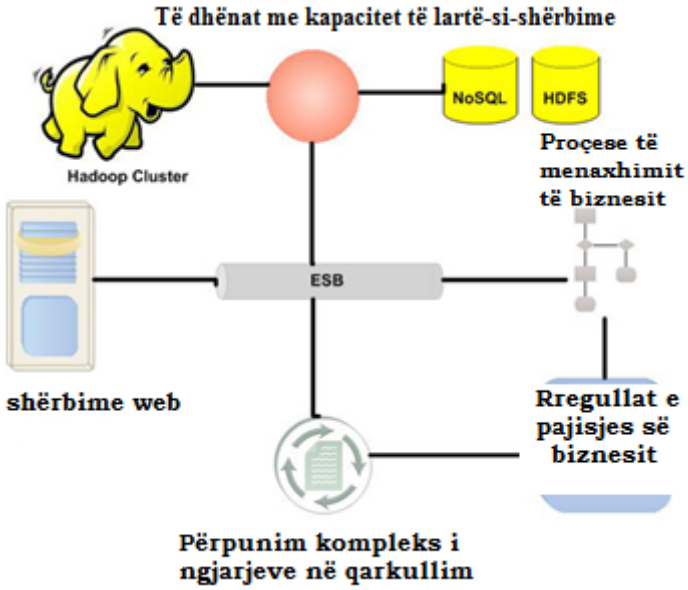
Ne mund ta parashikojmë BDaaS si një proçes i rrjedhjes së të dhënave nga lart-poshtë në një SOA. Në figurën 58, të dhënat me kapacitet të lartë rrjedhin në mbledhësit MapReduce dhe përpunohen nga CEP për tu pasqyruar si një shërbim në një platformë të shkallëzuar të një rrjeti virtual privat në një zonë lokale.

Një nga zgjidhjet më të përdorura gjerësisht të NoSQL është Apache Cassandra [39]. Facebook e ka origjinën nga Cassandra, i cili tani është në prodhim në Twitter, Reddit, Digg, dhe rrjete të tjera të mëdha.

Një grup Cassandra mund të ndërtohet me qindra nyje që përpunojnë petabajte të të dhënave dhe mund të integrohet me Hadoop, një burim i hapur MapReduce. Hadoop lejon vëllime të mëdha të të dhënave të përpunohen, duke i mbajtur të dhënat në grupin e tij origjinal.[43]Platformat e tjera Hadoop janë të rezervuara për data warehouse, Pig, për vendosjen e analizës së të dhënave, Chukwa, për monitorimin e sistemeve të shpërndara që përdorin HDFS, Flume për të grumbulluar të dhëna në lëvizje, dhe Sqoop për të lejuar transferimin e pjesës më të madhe të të dhënave midis Hadoop dhe bazave të të dhënave relacionale.Shkallëzimi elastik përmes virtualizimit cloud gjithashtu lehtëson të dhënat me kapacitet të lartë.

Një platformë e të dhënave me kapacitet të lartë përfshin masivisht përpunime paralele software që ekzekutohen në servera të shumtë, Petabyte – platforma të shkallëzuara që mbledhin, menaxhojnë dhe analizojë të dhënat e ndërmarrjeve, të shpërndara në ruajtjen hardware të të dhënave, në kujtesën e të dhënave, rrjetet e data-mining, dhe NoSQL-ruajtjen e të dhënave të shkallëzuara. Hadoop ishte kryesisht një përgjigje teknikë ndaj kufizimeve të arkitekturave të vjetra analitike dhe arkitekturave të ruajtjeve, të tilla si rrjetet e ruajtjes së zonës, bazat e të dhënave paralele, dhe mjeteve të inteligjencës së biznesit.

Zbatimet dhe pajisjet lidhëse mundësojë integrimin transaksionl të të dhënave dhe shërbimeve. Përshtatësit e shërbimeve Web lejojnë ndërfaqet e shërbimeve Web tek sistemet e aplikimit, bazës së të dhënave, ose platformës.



Figurë 58. SOA dhe të dhënat me kapacitet të lartë

Kompanitë janë gjithashtu duke zhvilluar mjeteve të mira për të kuptuar të dhënat me kapacitet të lartë të tilla si Splunk dhe Sumo Logic.

Implikimet për të ardhmen

Të dhënat me kapacitet të lartë, në kombinim me cloud, EDA, dhe SOA janë duke përcaktuar të ardhmen e teknologjisë së informacionit. Edhe pse forma e normalizuar e ruajtjes së të dhënave operative do të mbajë vlerën e saj, të dhënat me kapacitet të lartë jo vetëm që do të shtohen, por përfundimisht do të zëvëndësojnë skemat Star të të dhënave datawarehouse.

Konkluzione

Në mjedise ku aplikacioneve heterogjene lidhen me bazash të dhënash të shpërndara, është me rëndësi të veçantë integrimi i tyre, i cili është operacion i komplikuar dhe me kosto për biznesin.
Në këtë punim u trajtua roli i një shtresë ndërmjetëse e cila do të komunikojë me të gjithë databazat e shpërndara në rrjet duke ruajtur integritetin dhe performancën, pavarësisht nga specifikat apo numri i aplikacioneve heterogjene që ndodhen në këtë mjedis të shpërndarë.

Në punim u trajtuan mënyrat e ndryshme që egzitojnë për integruar aplikimet e veçanta në organizata mbi sistemet e bazave të dhënave, në një bashkësi aplikimesh bashkëpunuese , e cila sjell që zhvillimi i tyre të bëhet sa më fleksibël.

U trajtua roli databazave si një nga faktorët kryesorë që luan rol në këtë integrim, ku u shfaqën disa problematikat që lidheshin me komunikimin bazave e të dhënave të shpërndara (distributed) në rrjet.
Aksesimi i këtyre të dhënave lidhet direkt me komunikimin në rrjet,ku një nga pjesët e punimit që u trajtua ishte pikërisht studimi i performancës që ka rrjeti në ketë ambjent të shpërndarë dhe egzistojnë metoda që të optimizohet përdorshmëria e rrjetit duke përzgjedhë protokollet e rrugëzimit të cilat ndikojnë në eleminimin bllokimeve (starvation) në rrjet. Gjithashtu u vlerësua dhe u tregua që mund të arrihet një performancë më e mirë duke përdorur teknikën “Flow Label” e cila mundësohet në protokollin IPv6.

Një nga avantazhet më të mëdha që ofron ky model shpërndarjeje është që pavarësisht se të dhënat janë te shpërndara fizikisht, në nivelin llogjik të dhënat mund të aksesohen nga kudo.

Si rezultat i punimit u arrit që në një mjedis të shpërndarë bazash të dhënash të krijohej një shtresë ndërmjetëse e cila do të komunikojë me të gjithë databazat e shpërndara në rrjet duke ruajtur integritetin dhe performancën, pavarësisht nga specifikat apo numri i aplikacioneve heterogjene që ndodhen në këtë mjedis të shpërndarë.

Shtresa ndërmjetëse (middleware) u krijua sipas arkitekturës së mesazheve dhe radhëve (Message Queuing) të cilat përdorin parimin asinkron.
Kërkesat për ndryshime në bazat e të dhënave të shpërndara relacionale apo aplikacioneve heterogjene, u implementuan direkt në nivel databaze duke ruajtur pavaresinë nga njëra-tjetra.

Për të mundësuar komunikimin e shtresës ndërmjetëse me bazat e të dhënave të ndryshme dhe të shpërndara, u përdor standarti i XML për të mundësuar të dhënat shkëmbehen në trajtë mesazhesh ne format XML.
Formati XML është i njohur dhe i integruar në shumicën e sistemeve të menaxhimit të databazave, të cilat suportojnë ruajtjen e të dhënave në format XML dhe gjenerimin e XML duke u nisur nga të dhëna relacionale. Kalimi nga formati XML në të dhëna relacionale dhe anasjelltas do të jetë ishte pjese e punimit.

Teknikisht në këtë punim u përdor ambjenti i sistemit të menaxhimit të bazave të

dhënave Sql Server i cili suporton në brendësi arkitekturën e mesazheve dhe radhëve me anë të shërbimeve Broker të integruara në brendësi të tij.

U trajtua një rast studimi në një bankë:
Krijimi i një sistemi auditi i qendërzuar në një bankë, i cili arshivon gjithë ndryshimet që ndodhin nga aplikacionet kryesore të përdorura në bankë që ndodhen në servera të ndryshëm në rrjet duke mos ndryshuar kodin në aplikacionet heterogjene të përdorura, por duke e mundësuar këtë nëpermjet shërbimeve të mesazheve dhe radhëve broker direkt në bazë të dhënash.

Kërkesat e dërguara nga bazat e të dhënave në trajtën asinkrone, optimizojnë ndjeshëm kohën e përgjigjes të ç'do aplikacioni që është pjesë e rastit të studimit.

Disa nga problematikat e që u shfaqën lidhur me këtë rast studimi ishin:

- Rritja e log-ut audit në bazën e të dhënave qëndrore në mënyrë të pakontrolluar. Kur trendi i të dhënave në databazë rritet me kapacitete të larta, është u vu re një degradim i performancës.

  Si zgjidhje për këtë problem u përqendruam në teknikën e arshivimit, dhe particionimit, duke përfituar në :
  mbajtjen nën kontroll të dhënave dhe duke përfituar në performancë.

- Qëndrueshmëria ndaj dëmtimit për bazën e të dhënave.
  Në sistemin e të dhënave të studimit i cili bashkëvepron me databazat e tjera në rrjet, të dhënat ruhen në makina virtuale në server.Një ndër kërkesat e domosdoshme është qëndrueshmëria ndaj dëmtimit për bazën e të dhënave.

  Mqse të dhënat janë me kapacitete të larta dhe ndryshimet për këtë rast janë të shpeshta, u studiua teknika e Migrimit Live, e cila është dërgimi i shërbimeve dhe të dhënave nga një makinë fizike në një tjetër në kohë reale për të mundësuar qëndrueshmërinë ndaj dëmtimit.
  Faktor kryesor ishte performanca, dhe për këtë arsye u përqëndrova në optimizimin e teknikës së Migrimit Live.

Ky sistem i qendërzuar dhe i madh do të shërbejë në vazhdimësi si burim kryesor i teknikave data Mining që mundësohen në Sql Server me "Analysis Services", të cilat do të përdoren për të krijuar modele të gatshme dhe parashikuese për auditorët e ndryshëm.

Referencat

[1] Service-Oriented Architecture: Concepts, Technology, and Design – Thomas Erl Pretence ,2005

[2] Web services and Enterprise Application Integration - Chris Wong

[3] Elements of a Successful Enterprise Application Integration (Artikull Syntel)

[4] Guidelines for Application Integration (Patterns & Practices) - Microsoft Press 2003 Microsoft Corporation Version 1

[5] Web Services – An Instrument to Resolve the Problems of Information Systems Integration – G. Popova (2006)

[6] Diploma "An Evaluation of Enterprise Application Integration Solutions and the Role of Web Services" - Ulrich Endlich (2005)

[7] Application Integration Using Oracle 9i - Suresh Kumar Neti, Sierra Atlantic Software Services Ltd.

[8] Information Systems Integration and Interoperability - Mattias Jansson (2007)

[9] Enterprise Application Integration using MQSeries and Web services - Evan Mamas

[10] Enterprise Application Integration - Strategies for the Telecommunication Market - Martin Rosendahl and Erik Rune

[11] Common Object Request Broker Architecture - for embedded: core CORBA/e specification (2006)

[12] Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions - Gregor Hohpe , Bobby Woolf : Addison-Wesley Professional (2003)

[13] Enterprise Architecture for Integration Rapid Delivery Methods and Technologies: Clive Finkelstein (2006)

[14] Professional J2EE EAI - Matjaz Juric, Ramesh Nagappan, S. Jeelani Basha, Rick Leander, Peter Long (2001)

[15] Analyses of QoS Routing Approach and the Starvation's Evaluation in LAN. Ariana Bejleri, Igli Tafaj, Aleksander Biberaj, Ermal Beqiri, and Julian Fejzaj. BCI Local, CEUR-WS.org, (2012)

[16] The Role of the Adaptive Network in Service-Oriented Architectures A META Group White Paper 800- 945-META [6382],2005

[17] Database System Concepts, sixth edition,Abraham Silberschatz,Henry F. Korth,S. Sudarshan, 2011

[18] http://www.infosys.com/IT-services/independent-validation-testing-services/white-papers/Documents/modern-application-testing.pdf

[19] http://www.nevatech.com/blog/post/2013/07/29/SOA-Governance-Service-Virtualization

[20] http://www.mitre.org/sites/default/files/pdf/09_0743.pdf

[21] The improvement of Live Migration in Data. Center in different virtual environment. Julian Fejzaj, Igli Tafa,Elinda Kajo, IJCSI November 2012

[22] The Evaluation of Performance in Flow Label and Non Flow Label Approach Based on IPv6 Technology. Ariana Bejleri, Igli Tafaj, Ermal Beqiri, Julian Fejzaj, Aleksander Biberaj BCI (Local) 2012: 1-4

[23] Enterprise SOA: Service-Oriented Architecture Best Practices By Dirk Krafzig, Karl Banke, Dirk Slama, 2004

[24] Pro SQL Server 2008 Service Broker Copyright © 2008 by Klaus Aschenbrenner

[25] Nëntor 2012, Fejzaj J. & Xhina E , "Modelimi i Arkitekturës së Orientuar drejt shërbimeve në Sql Server 2008 duke përdorur Service Broker në " Buletinin e Shkencave Natyrore BSHN Vol 13"

[26] David Campbell, "Service Oriented Database Architecture: App Server-Lite?" Microsoft Research (September 2005)

[27] http://technet.microsoft.com/library/Cc966516

[28] Pro SQL Server 2005 Service Broker by Klaus Aschenbrenner,2007

[29] http://msdn.microsoft.com/en-us/library/ms954587.aspx

[30] Service Broker - the asynchronous messaging framework in SQL Server 2005 by Klaus Aschenbrenner,2010

[31] http://technet.microsoft.com/en-us/library/bb522493(v=sql.105).aspx

[32] https://sites.google.com/a/ictedu.info/sql2008/creating-xml

[33] https://sites.google.com/a/ictedu.info/sql2008/creating-xml#toc-xml-2-table

[34] https://www.simple-talk.com/sql/learn-sql-server/sql-server-xml-cribsheet/

[35] Pro SQL Server 2008 XML, Michael Coles , 2008

[36] http://www.sqlservercentral.com/articles/OpenXML/usingopenxml/1881/

[37] http://www.scaledb.com/large-database.php

[38] Fejzaj J. & Xhina E. & Saatciu D. & Bimbari B,"Performance improvement through table partitioning (comparison of table partitioning in SQL server 2008)" Journal "*Science, Innovation and New Technology*",Vol. 1, No. 1, 2011

[39] Philip Wik, Cassandra and Big Data", DataCenterAcceleration", September 18, 2012: http://www.

[40] datacenteracceleration.com/author.asp?section_id=2432&doc_id=250926&

[41] http://www.servicetechmag.com/i70/0213-1

[42] Manoj Mansukhani,Service Oriented Architecture White Paper (2005)

[43] https://developer.yahoo.com/hadoop/tutorial/

[44] Analysis of Data mining Application,Bora Bimbari,Dhimiter Kikina, Adriola Faqolli, Julian Fejzaj

Përmbledhje

Me qëllim të bashkëveprimit të aplikacioneve heterogjene, të cilët përdorin bazat e të dhënave të shpërndara në rrjet, është me rëndësi të veçanë përzgjedhja e modelit arkiteturor, i cili ndikon drejtëpërdrejtë në koston dhe vendimarrjen për sistemet IT. Service Oriented Architecture (SOA) është një model i tillë në infrastrukturën IT, i cili proceset e biznesit i paketon si shërbime të pavaruara që mundësojnë aplikime të ndryshme të shkëmbejnë të dhëna me njëra-tjetrën duke ofruar gjithashtu një fleksibilitet për integrimin e sistemeve të vjetra në ato të reja.

Në këtë punim, fillimisht do të studiohen bazat e të dhënave për mënyrën e integrimit të koncepteve SOA në brendësi të databazës dhe mjetet që e mundësojnë këtë, koncept i njohur ndryshe si Service Oriented Database Architecture(SODA).

Si konkluzion është menduar të ndërtohet një aplikacion 'middleware' sipas arkitekturës së shërbimeve që dërgon të dhëna në formën e mesazheve dhe radhëve (Message Queuing), që përdorin parimin asinkron, i cili do të shërbejë si shtresë ndërmjetëse për transferimin e të dhënave të aplikacioneve të ndryshme nga disa databaza të shpërndara në rrjet, pa ndryshuar logjikën e programimit të aplikacioneve që lidhen me to. Kjo do të bëhet e mundur në sistemin e menaxhimit të bazave të dhënave Sql Server me anë të shërbimeve Broker, të cilat mundësojnë një shkëmbim fleksibël midis shërbimeve të ndryshme të shpërndara në rrjet, bazuar në modelin e mesazheve dhe radhëve të të dhënave të konvertuara në formatin standard XML

Aplikacioni është menduar si një sistem auditi i qendërzuar në një bankë që mbledh gjithë ndryshimet që ndodhin në databazat e shpërndara në rrjet, do të shërbejë si një burim kryesor informacioni për të trajtuar disa çështje të veçanta që dalin nga ky bashkëveprim si: performanca e shkëmbimit të të dhënave, rritja në kapacitete të mëdha e të dhënave dhe optimizimi i kërkesave, ndikimi i infrastrukturës së rrjetit si dhe metodat dhe optimizime të cilat lidhen me qëndrueshmërinë ndaj dëmtimit të bazave të të dhënave.

***Fjalët kyçe**: Arkitekturat e Bazave të Dhënave të Orientuara nga Shërbimet,Aplikacione Heterogjene, XML, Shërbime Broker, Databazat e Shpërndara*

Abstract

Heterogeneous applications are applications that use databases distributed in the network they operate. In order for them to cooperate, the architecture model selection is very important, as it directly impacts the decision making and costs of IT systems. Service Oriented Architecture (SOA) is a model in the IT infrastructure that packages business processes as independent services. Services enable different and independent applications to exchange data, and provide flexibility in integrating the old systems into the new ones.

Service Oriented Database Architecture (SODA) integrates SOA concepts in the domain of databases and related management systems. Such architecture is based on the paradigm of sending data using a Message Queuing model following asynchronous message exchange principles, and then transferring the data of different applications from distributed databases across the network without changing the programming logic of the linked applications.

The purpose of this thesis is to first study the theoretical background of SODA. Following its principles, a middleware application is built, which implements a flexible mechanism for data exchange between different services distributed over the network, based on the Message Queuing model of data converted in XML standard format. The application makes use of Microsoft SQL Server management system, and of its Service Broker.

This application is thought as an audit system centered on a Bank that collects all the changes that happen in the network distributed databases. Its purpose is to serve as the main source of information to treat special and critical issues as: data exchange performance, large increase in capacity of data, query optimization, network infrastructure influence, methods and optimizations related to the consistency of databases failure.

Keywords: *Service Oriented Database Architecture, Heterogeneous Applications, XML, Service Broker, Distributed Databases*