Razvoj procedure za oporavak baze podataka od nepredviđenih situacija

Stupar, Marko

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Marko Stupar

Razvoj procedure za oporavak baze podataka od nepredviđenih situacija

Završni rad

Pula, rujan, 2020.

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Marko Stupar

Razvoj procedure za oporavak baze podataka od nepredviđenih situacija

Završni rad

JMBAG: 0303046281, izvanredni student

Student: Marko Stupar, izvanredni student

Kolegiji: Baze podataka 1

Studijski smjer: Informatika

Mentor: izv. prof. dr.sc. Tihomir Orehovački

Pula, 22. rujan, 2020. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Marko Stupar, kandidat za prvostupnika Informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

Marko Stupar

U Puli, 22. rujna, 2020. godine

IZJAVA

o korištenju autorskog djela

Ja, Marko Stupar dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom „Razvoj procedure za oporavak baze podataka od nepredviđenih situacija“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 22. rujna, 2020. godine

Potpis

Marko Stupar

SADRŽAJ

UVOD

Temelj svakog poslovnog sustava odnosno svake kompanije je kvalitetan i razrađen informacijsko komunikacijski sustav. Njegova uloga je da konstantno ili na zahtjev opskrbljuje potrebnim informacijama sve razine upravljanja i odlučivanja. Dakle suština mu je informacija, te kao takva ona je neizmjerno bitna unutar samog sustava.

Upravo radi toga, gledajući sa tehničkog stajališta, unutar modernih informacijsko komunikacijskih sustava postoje sustavi zaštite same informacije. Jedan od sustava zaštite je i sigurnosno kopiranje podataka (eng. „Backup“). Sigurnosna kopija podataka se najčešće sprema na neke vanjske medije ili virtualizirane udaljene oblake. Osim same informacije koju možemo promatrati kao nešto apstraktno i neopipljivo te razno raznih programskih rješenja koja služe za pohranu ili pristup samim informacijama, unutar informacijsko komunikacijskog sustava postoje i fizičke komponente koje omogućuju i pružaju usluge. Fizičke komponente kao takve se najčešće postavljaju i grupiraju unutar jedne ili više prostorija te predstavljaju srce samog informacijsko komunikacijskog sustava (eng. „server room“).

Dodatkom fizičkih komponenti pojavljuju se novi rizici i opasnosti za sami informacijski sustav. Naime, dolazi u pitanje i sama fizička sigurnost sobe gdje se poslužitelji nalaze, također dolazi do moguće prijetnje od loših vremenskih uvjeta. Na osnovi toga unutar svakog informacijskog sustava se mora razraditi plan za oporavak od nepredviđenih situacija (eng. „Disaster recovery plan“). Taj plan se sastoji od točnih procedura koje treba provoditi konstantno ili po potrebi kako bi spriječili gubitak usluga ili podataka. Sam plan je veoma kompleksan te je najčešće sastavljen od više mogućih loših scenarija te detaljno opisuje što napraviti za određenu komponentu samog sustava.

U ovom završnom radu fokusirati ćemo se samo na dio plana za oporavak od nepredviđenih situacija odnosno razvitak procedure za oporavak baze podataka kao komponente informacijskog sustava. Pokazat ćemo i objasniti okolinu i tehnologiju koja se koristila pri izradi(poslužitelj, operacijski sustav, verzija baze i alati).

1. POSTAVLJANJE OKOLINE INFORMACIJSKOG SUSTAVA I OBJAŠNJENJE OSNOVNIH POJMOVA I ALATA

Da bi detaljno opisali i shvatili razvitak procedure za oporavak baze podataka od nepredviđenih situacija moramo najprije pojasniti osnovne pojmove. Počevši od postavljanja okoline unutar koje će raditi naša baza podatka i sve ostale komponente. Kroz sljedećih nekoliko poglavlja ovog završnog rada ćemo objasniti i detaljnije opisati sve hardverske i softverske komponente našeg informacijskog sustava koje sudjeluju u našoj proceduri.

1.1. Soba s poslužiteljima – hardver komponente

Organizacije i tvrtke najčešće započinju svoju IT praksu tako što su im poslužitelji postavljeni ispod stolova ili u improviziranim ormarima. U njima se osim samih poslužitelja može nalaziti i razno razna mrežna oprema, oprema za pohranu podataka ili oprema za sigurnosnu kopiju podataka. Kako organizacija ili tvrtka raste tako raste i njeno oslanjanje na podatke iz samog informacijskog sustava, shodno tome dolazi do potrebe da se osigura stabilnija i sigurnija okolina za kritične komponente samog informacijskog sustava. Bilo da je strah od prekida koji usporava produktivnost, gubitak podataka koji bi mogao naštetiti percepciji organizacije ili regulatorna usklađenost, IT osoba ili grupa prisiljeni su stvoriti prikladnije okruženje. Soba s poslužiteljima dakle predstavlja prvi korak organizacije ili tvrtke ka ozbiljnijem informacijskom komunikacijskom okruženju unutar radnog mjesta. U praksi osnovna poslužiteljska soba ima kontrolirano hlađenje te dva ili tri komunikacijska ormara za poslužitelje, mrežnu opremu, opremu za pohranu i opremu za sigurnosnu kopiju podataka.

Također, u nekom većem obimu, postoje organizacije ili tvrtke koje imaju velike udaljene lokacije, na kojima se može smjestiti na stotine zaposlenika i kojima je to potrebno radi lokalne obrade podataka, komunikacijskih usluga, dijeljenja datoteka i općenito brzog odaziva sustava kojem se pristupa. Često takve organizacije šire svoje poslovanje na globalnu razinu te samim time to zahtjeva regionalne urede smještene

u nekim stranim zemljama. Takva udaljena mjesta odnosno uredi zahtijevaju i svoje lokalne poslužitelje kako bi osigurali visoku dostupnost i sigurnost aplikacija koje se koriste. Na preuzetoj slici u nastavku vidim primjer jedne poslužiteljske sobe.

Slika 1. Primjer poslužiteljske sobe (*Server room example,* 2017)

1.1.1. Mrežna oprema

Mrežna oprema služi za umrežavanje, odnosno elektroničko povezivanje radi razmjene informacija. Resursi unutar informacijskog sustava kao što su datoteke, aplikacije i pisači su često dijeljeni unutar same mreže. Prednost samog umrežavanja može se jasno vidjeti u aspektima sigurnosti, učinkovitosti, upravljivosti te isplativosti. Umreženost također omogućuje suradnju između korisnika samog informacijskog sustava. Sama mreža se može sastojati od više hardware-skih komponenti, neke od njih su:

- Pojedinačna točka mreže (eng. node) - (klijent, računalo, poslužitelj…) – hardware-ska komponenta koja je dio mreže, u njoj sudjeluje kao klijent.

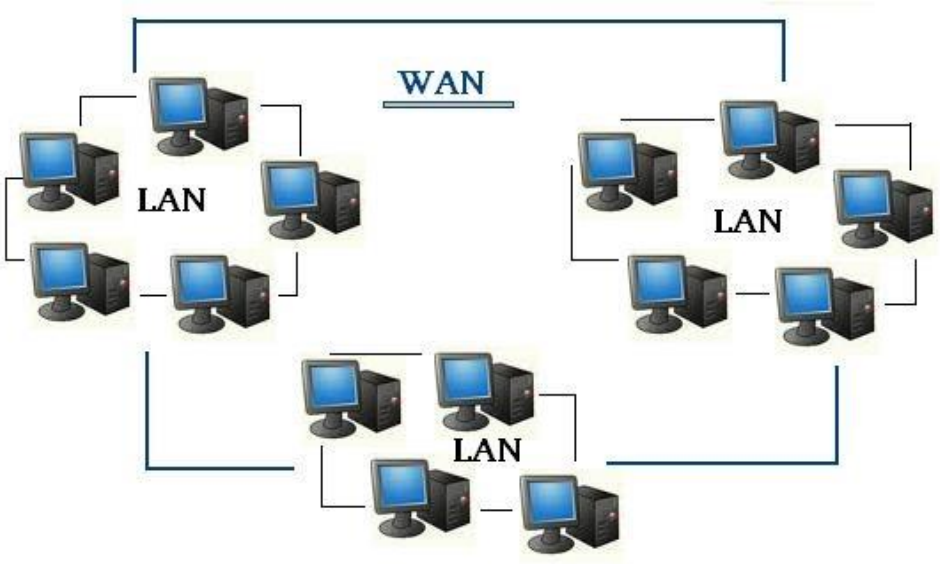
- Mrežni prekidač, preklopnik (eng. switch) – hardware-ska komponenta koja je dio mreže, uređaj koji povezuje ostale uređaje. Upravljaju protokom podataka kroz mrežu prenoseći primljeni mrežni paket samo na jedan ili više uređaja kojima je paket namijenjen. Svaki drugi mrežni uređaj povezan na preklopnik može se prepoznati po svojoj mrežnoj adresi, što samom mrežnom preklopniku omogućuje da usmjeri protok prometa maksimizirajući sigurnost i učinkovitost mreže.

Slika 2. Prikaz mrežnog prekidača ili preklopnika (*Cisco ethernet switch,* bez dat.)

- Mrežni usmjerivač (eng. router) – hardware-ska komponenta koja je dio mreže, uređaj koji prosljeđuje pakete između više računalnih mreža. U praksi se najčešće koristi za usmjeravanja prometa na internetu, podaci poslani putem interneta su u obliku podatkovnih paketa. Paket se prosljeđuje od jednog usmjerivača do drugog kroz razne računalne mreže dok ne dosegne odredišni čvor.

Također, valja napomenuti da je mnogo vrsta mreža dostupno u mrežnim industrijama, a najčešće su:

- Lokalna mreža (eng. LAN – Local area network) – oblik mreže koja se sastoji od dvije ili više točaka (eng. nodes) povezanih na kratkoj udaljenosti. Primjer iz prakse bio bi kompletna računalna mreža jedne poslovne zgrade uključujući poslužiteljske sobu i sve klijente koji su povezani na nju.
- Mreža na širem području (eng. WAN – Wide area network – oblik mreže koji pokriva šire područje od LAN-a, područje između više zgrada, gradova i slično. Najlakše ju je shvatiti tako da se zamisli da ako se više glavnih manjih LAN-povežu međusobno tvore jednu WAN mrežu.



Slika 3. Prikaz jednostavne LAN i WAN računalne mreže(*LAN and WAN,* bez dat.)

1.1.2. Poslužitelji

Govoreći unutar spektra informacijsko komunikacijskih tehnologija poslužitelj je specijalizirano računalo koje pruža funkcionalnost za druge programe ili uređaje koji se nazivaju klijenti. Takav model arhitekture nazive se klijent-poslužitelj. Poslužitelji mogu pružiti razne „usluge“ poput dijeljenja podataka ili resursa između više klijenata, ili obavljanja raznih proračuna. Povećane hardverske specifikacije i pouzdanost razlikuje poslužiteljsko računalo od običnih klijentskih računala. Sami poslužitelj može vršiti više zadaća unutar cjelokupnog informacijskog sustava, neki od oblika u kojem se koriste poslužitelji su:

- Aplikacijski poslužitelj (eng. application server) – Poslužuje web aplikacije koje klijenti izvode putem web preglednika
- Poslužitelj za virtualizaciju (eng. virtualization host, hypervisor) – Poslužitelj čiji se računalni resursi virtualiziraju, te se unutar njega stvara više manjih virtualnih poslužitelja. Svaki od tih manjih poslužitelja obavlja svoju zadaću, neovisno jedan o drugome
- Poslužitelj datoteka (eng. file server) – Poslužitelj koji služi za dijeljenje raznih datoteke i mapa unutar računalne mreže

- Poslužitelj za ispis (eng. print server) – Poslužitelj koji služi za dijeljenje pisača unutar računalne mreže, klijenti potom koriste pisače za ispis
- Poslužitelj baze podataka (eng. database server) – Poslužitelj koji služi za pohranu i dijeljenje bilo kakvog oblika baze podataka unutar računalne mreže

Slika 4. Prikaz poslužitelja, Sun Sparc T5-2(*Oracle SPARC T5-2 Server,* bez dat.)

1.2. Programska komponenta – softver

U nekoliko prethodnih poglavlja ovog završnog rada opisali smo hardver komponente unutar samog informacijskog sustava, u nastavku ćemo pobliže opisati i programske komponente tkz. softver, počevši od pojma virtualizacije poslužitelja. Također ćemo objasniti što je to operacijski sustav, što je baza podataka te kako ona funkcionira. Osim toga spomenuti ćemo i opisati sve alate unutar operacijskog sustava koje ćemo koristiti kod naše procedure za oporavak.

Softver je u osnovi zapravo zbirka podataka ili računalnih uputa koje računalu odnosno poslužitelju govore kako i što treba raditi. Po definiciji iz računalne znanosti softverom se mogu smatrati svi podaci i informacije koje obrađuje računalni sustav. Sam softver se može pojaviti u dva najosnovnija oblika:

- Sistemski softver (eng. system software) – niska razina softvera, koristi se najčešće za upravljanje samim hardverom. Sastoji se od uputa za strojni jezik

podržanog od strane pojedinog procesora. Neki oblici softvera niske razine su operacijski sustavi i upravljački programi.

- Aplikacijski softver (eng. application software) - viša razina softvera, softver koji koristi računalni sustav za obavljanje posebnih funkcija izvan osnovnog rada samog računala

1.2.1 Virtualizacija poslužitelja

Virtualizacija poslužitelja je tehnologija koja omogućuje stvaranje virtualnih okruženja, unutar kojih se omogućuje pokretanje više različitih aplikacija. Na taj način nastaju zasebni manji poslužitelji od kojih svaki kao što smo prethodno naveli može izvršavati svoju zadaću. Sama virtualizacija se postiže kreiranjem virtualnog računala ili operacijskog sustava koji se u suštini ponaša kao pravi, ali nije. Primarni cilj virtualizacije je zapravo izolacija i podjela radnog opterećenja. Postoji više oblika same virtualizacije kao što su:

- OSV model (eng. operating system virtualization) – model koji se bazira na virtualizaciji unutar samog operacijskog sustava. Svaka virtualizirana okolina sadrži svoju „kopiju“ operacijskog sustava, takve okoline nazivaju se još i kontejneri (eng. containers)
- VM model (eng. virtual machines) – model kod kojega više različitih instanci operacijskih sustava rade na jedno te istom poslužitelju te koriste njegove resurse. Svaka virtualizirana mašina je zasebna te koristi svoj dio dostupnih resursa. Sami poslužitelj na kojem se vrši virtualiziranje naziva se još i hipervizor.

Također bitno je navesti kako postoje razne kompanije na tržištu koje nude svoje tehnologije virtualizacije, neke od njih su VMware (ESXi) koji je i predvodnik industrije i najzastupljeniji po podatkovnim centrima diljem svijeta, osim njega postoje još i Microsoft (HyperV), Citrix(Xen), RedHat(KVM), Oracle (VM for SPARC) i mnogi drugi. Upravo ćemo se u nastavku ovog završnog rada fokusirati i pobliže objasniti Oracle-

ov „VM for SPARC“ jer ćemo upravo to tehnologiju virtualizacije koristiti kod razvitka naše procedure za oporavak od nepredviđenih situacija.

1.2.1.1 Logičke domene

Kod virtualizacije „VM for SPARC“ koristi se mogućnost kreiranja tkz. SPARC virtualnih mašina (eng. logical domains). Ova tehnologija hipervizora omogućava rad svih kreiranih logičkih domena s manje dodatnih hardverskih troškova (eng. overhead) za razliku od klasičnih modela virtualizacije. Kreiranim logičnim domenama je znatno olakšan pristup fizičkim resursima samog poslužitelja, kao što su procesor, radna memorija i mjesta za pohranu podataka. Idealno je tehnološko rješenje za konsolidiranje više virtualnih Oracle Solaris virtualnih mašina koje trebaju imati izolirana i različita okruženja.

Upravo to izolirano okruženje je najbitnija osobina „VM for SPARC“ virtualizacijske tehnologije. Svaka instanca logičke domene pokreće svoju verziju operacijskog sustava Oracle Solaris, shodno tome svaki operativni sustav unutar sebe može imati svoje aplikacije, dokumente i slično. Svi oni zajedno rade na istom poslužitelju bez ikakvih konflikata, pojedina domena se može neovisno o drugima zaustavljati i ponovo pokretati. Veoma je istaknuta i robusnost koja se manifestira u činjenici da sistemska greška, prisilno zaustavljanje i ponovno pokretanje jedne logičke domene nema nikakvog utjecaja na druge logičke domene koje rade na istom poslužitelju.

Na slici broj pet u nastavku vidimo kako izgleda prikaz unutar komandne linije kada izlistamo sve dostupne logičke domene unutar jednog poslužitelja. Možemo primijetiti da postoje dvije logičke domene, svaka ima svoje dodijeljene resurse te stanje u kojem se nalazi, „primary“ je aktivna i upaljena, dok je „ldom1“ ugašena. „Bound“ stanje znači da smo osigurali računalne resurse te je „ldom1“ spreman za pokretanje.

```
# ldm list   #(NOTE: this was done after bind, before start)
NAME         STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary      active     -n-cv-  SP      4     3G       1.2%  38m
ldom1        bound      ------  5000    4     1G
```

Slika 5. Prikaz liste logičkih domena iz komandne linije (Victor i sur., 2017)

Možemo zaključiti da su logičke domene zapravo virtualizirane putem VM modela virtualizacije. Dakle, svaka je zasebna virtualna mašina, međutim što ako radi potreba informacijskog sustava treba raditi više različitih aplikacija unutar iste logičke domene? U nastavku ćemo dignuti virtualizaciju još jednu razinu iznad, ući ćemo u sloj operacijskog sustava, opisati ćemo virtualne zone, odnosno manje kontejnere koje rade unutar virtualiziranih logičkih domena.

1.2.1.2. Virtualne zone (Oracle Solaris Containers)

Prethodno smo naveli OSV model, odnosno kontejnerizaciju unutar operacijskog sustava, u ovom poglavlju ćemo ju detaljnije opisati radi potrebe shvaćanja procedure za oporavak baze podataka od nepredviđenih situacija.

Virtualizacija na razini operacijskog sustava je praktički bila nepoznata kada je 2005. godine Solaris 10 izašao na tržište. Revolucionirao je pojam kontejnerizacije tako što je uveo model gdje se virtualizacija vrši tako što operacijski sustav kreira virtualnu softversku okolinu unutar koje rade određene aplikacije. Ta okolina naizgled radi kao zasebna instanca operacijskog sustava, izolirana je, međutim u pozadini ona dijeli operacijski sustav i kernel sa samim originalnim poslužiteljem (operacijskim sustavom). Iako su virtualne zone u pozadini malo drugačije okruženje od običnog operacijskog sustava Solaris, osnovno načelo dizajna je softverska kompatibilnost. Što u globalu znači da bilo koja aplikacija koja radi na bazičnom operacijskom sustavu će također raditi i na virtualiziranoj zoni, i unatrag. Također, zone imaju svoj vlastiti životni ciklus koji je neovisan o osnovnom operacijskom sustavu, mogu se pokretati i koristiti kao zasebni sustavi. Gledajući sa aspekta računalnih zahtjeva, zone su veoma lagane i ne zahtijevaju puno resursa, dio toga proizlazi iz činjenice da nisu čitav operativni

sustav, ne testiraju hardver na kojemu rade, ne učitavaju kernel ili polazne programe za uređaje (eng. device drivers), osnovni operacijski sustav koji je u pozadini čiti sve to umjesto njih. Danas najpopularniji slučaj takve tehnologije je Docker.

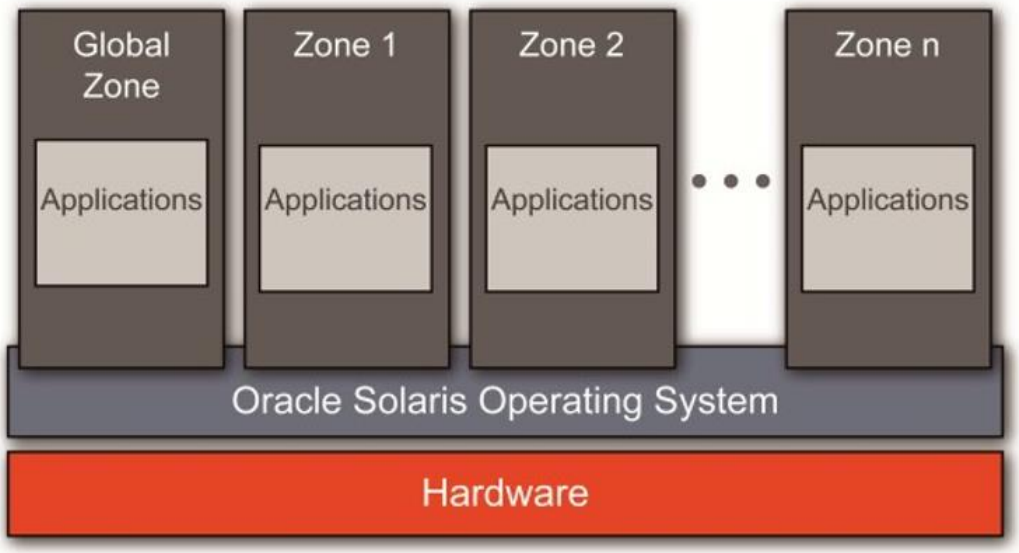
Ako se vratimo na našu okolinu, dakle, unutar logičke domene, koja je kao što smo već spomenuli zasebna virtualna mašina kreiramo virtualnu zonu. Virtualna zona je ništa drugo nego jedna konfiguracijska datoteka unutar koje odredimo sve parametre koje želimo da zona poprimi.

Na slici broj šest možemo vidjeti kako izgleda ispis na komandnoj liniji svih virtualnih zona unutar jedne logičke domene. Zona „global“ je pokrenuta, dok sa druge strane zona „myzone“ ima učitanu konfiguraciju i spremna je za pokretanje.

```
GZ# zoneadm list -cv
  ID NAME     STATUS     PATH                   BRAND    IP
   0 global   running    /                      native   shared
   - myzone   configured /zones/roots/myzone  native   shared
```

Slika 6. Prikaz liste virtualnih zona unutar jedne logičke domene (Victor i sur., 2017)

Na slici broj sedam se jasno vidi odnos između hardvera i raznih slojeva virtualizacije. Možemo primijetiti da imamo goli hardver na kojeg se putem hipervisora veže sloj VM model virtualizacije gdje „Oracle Solaris Operating System“ predstavlja jednu logičku domenu, na sloju iznad nje se još dodatno metodom OSV virtualiziraju zone. Kao što smo naveli svaka zona je priča za sebe, ima svoju mrežnu adresu i ime, te se zapravo predstavlja kao zasebni operacijski sustav. Operacijski sustav koji unutar sebe ima alate koje ćemo intenzivno koristiti kod naše procedure za oporavak baze podataka od nepredviđenih situacija, u sljedećim poglavljima ćemo pobliže opisati operacijski sustav Solaris, komandu ljusku „Bash“ te alate unutar nje kao što su „crontab“ i „rsync“.



Slika 7. Prikaz odnosa poslužitelj-logička domena-zone(*Zone Clusters Whitepaper,* 2011)

1.2.2. Oracle Solaris operacijski sustav

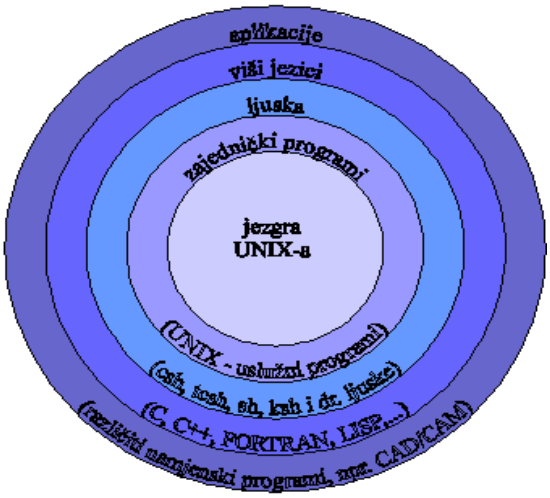
Iz literature (Bill Calkins, 2013) možemo iščitati da je Solaris operacijski sustav koji je nastao 1993. godine, stvorila ga je tvrtka „Sun Microsystems“. Godine 2010. Sun je kupljen od strane Oracle-a te Solaris mijenja ime u „Oracle Solaris“.

Baziran je UNIX operacijskom sustavu, što znači da je višeprogramski i višekorisnički sustav opće namjene. Poznat je po svojoj skalabilnosti, posebno kada se nalazi na SPARC poslužiteljima, neke od revolucionarnih značajki su mu ZFS i DTrace. Kao i svaki drugi UNIX sustav Solaris također ima svojstva u nastavku:

- Hijerarhijska organizacija datoteka u obliku stabla (eng. tree structure)
- Sve je datoteka
- Ulančavanje procesa
- Korištenje ljuske kao veze između korisnika i sustava, korisnik upisuje komande, sustav odgovara te izvršava zadane zadatke

Na slici broj osam vidimo prikaz slojeva operacijskog sustava unix, gdje je polazište svega jezgra ili tkz. kernel. Nakon nje idu uslužni odnosno zajednički programi te naposljetku ljuska pomoću koje korisnik „razgovara“ sa sustavom te mu daje

instrukcije. Iznad jezgre se mogu pojaviti još i programski jezici više razine (C,C++…) ili neki namjenski programi odnosno aplikacije.



Slika 8. Prikaz strukture UNIX-a (Žagar, 1995)

Ne ulazeći u dubinu i kompleksnost samog operacijskog sustava te kako on funkcionira, fokusirati ćemo se na ljusku. Točnije „Bash ljusku koja je posrednik između sustava i korisnika te sve alate koji će nam biti potrebni unutar nje.

1.2.2.1 Bash ljuska – Bash shell

Kao što smo već naveli pod definiciji (Arnold Robbins i dr., 2005) ljuska je spona između korisnika i operacijskog sustava, u suštini to je oblik programskog komandnog jezika s varijablama, kontrolnim naredbama i potprogramima. Djeluje kao tumač naredba (eng. interpreter) odnosno pročita liniju teksta, interpretira je i poduzme sve potrebne akcije za njeno provođenje. Kad je željena naredba izvršena ljuska vrati informaciju korisniku da je spremna prihvatiti sljedeću naredbu.

Konkretno BASH ljuska nastala je krajem osamdesetih godina prošlog stoljeća, zamijenila je to tada masovno korištenu „Bourne“ ljusku. Ubrzo nakon nastanka počela

se primjenjivati u UNIX operacijskim sustavima, danas je najpopularnija ljuska te se koristi kao zadana ljuska na više-manje svim UNIX distribucijama uključujući i Solaris operativne sustave. Kroz godine razvoja BASH je poprimio puno novih značajki, jedna od tih nama ključnih značajki je upravo mogućnost skriptiranja. Pokretanje unaprijed napisanih skripti koje odrađuju željene zadatke u točno zadanim intervalima je zapravo suština naše procedure za oporavak baze podataka od nepredviđenih situacija. Unutar spomenutih skripti su napisane linije naredbi, te naredbe pozivaju potprograme odnosno alate koji izvršavaju željene zadaće. Također valja napomenuti da svaka naredba odnosno potprogram koji pozovemo prilikom interakcije sa ljuskom unutar UNIX operacijskog sustava može imati i dodatne argumente, ti argumenti određuju odnosno mijenjaju svojstva naredbe koja se šalje prema sustavu. U nastavku ovog završnog rada opisati ćemo najbitnije alate koje ćemo koristiti unutar naših skripti, opisati ćemo čemu služe i koja je njihova glavna zadaća.

1.2.2.2. TAR naredba

Naredba „tar“ se unutar Solaris-a koristi za stvaranje arhivskih datoteka (eng. tarballs). Kao što smo prethodno naveli naredbi se mogu dodati argumenti kako bi se pobliže i detaljnije opisao željeni ishod naredbe, neki od argumenata izvađenih ih priručnika („TAR manual pages“, 2020) koji ćemo koristiti u nastavku prilikom prikaza procedure za oporavak baze podataka su:

- -c – stvori novi arhiv ili pregazi već postojeći
- -v – Radi verbose funkciju, odnosno prikaži postupak arhiviranja uživo na ekranu dok se provodi
- -f – Dopušta korisniku da specificira ime arhive koja će nastati

1.2.2.3. SCP naredba

SCP (eng. secure copy) naredba omogućuje kopiranje datoteka između dva poslužitelja na računalnoj mreži. Za prijenos podataka koristi „ssh“ protokol, odnosno koristi istu provjeru autentičnosti i pruža istu sigurnost kao i „ssh“.

1.2.2.4. RSYNC naredba

RSYNC (eng. remote sync) je brz i svestran alat za kopiranje datoteka. Omogućuje korisniku da kopira datoteke sa i na lokalnog poslužitelja. Sami alat nudi velik broj opcija i argumenata koji kontroliraju svaki aspekt njegovog ponašanja i omogućuju vrlo fleksibilne specifikacije zadaće koju želimo da rsync obavi. Jedna od bitnih značajki mu je i ta da koristi tkz. „delta“ prijenos odnosno smanjuje količinu podataka poslanih putem računalne mreže tako što šalje samo razlike između izvornih datoteka i postojećih datoteka na odredištu. U globalu najviše se koristi za izradu sigurnosnih kopija datoteka ili zrcaljenje pojedinih mapa sa mapama na drugim poslužiteljima na udaljenim lokacijama. Argumenti naredbe pronađeni u priručniku RSYNC („RSYNC manual pages“, 2020) koje ćemo u našem slučaju koristiti su:

- -a – Arhiva model, korisnik daje instrukciju rsync naredbi da zadrži sva svojstva datoteka koje kopira, a to su mekane poveznice(eng. soft links), prava pristupa, vremena izmjene, pripadnost grupi te vlasnike datoteka
- -v – Radi „verbose“ funkciju – prikazivanje postupka kopiranja na ekran
- -H – naredba će prilikom kopiranja datoteka zadržati i tvrde poveznice (eng. hard links)

1.2.2.5. SQLPLUS alat

SQLPLUS je interaktivni alat unutar komande linije za interakciju sa bazom podataka. Dolazi u paketu sa instalacijom Oracle baze podataka. Kod ovog alata je specifično to što ima svoje naredbe i okruženje putem kojeg pruža pristup bazi podataka te omogućuje korisniku unos i izvršavanje SQL upita i PL/SQL programskog koda. Neke od mogućnosti su mu:

- Formatiranje prikaza na ekranu, izvršavanje izračuna i pohrana rezultata upita u datoteku
- Ispis definicija tablica i objekata unutar baze podataka
- Razvitak i pokretanja raznih skriptnih datoteka
- Administriranje baze podataka

1.2.2.6. CRONTAB alat

Crontab alat odnosno datoteka sadrži instrukcije koje govore cron procesu da pokrene određenu akciju u određenom trenutku. Svaki korisnik unutar samog operacijskog sustava definira svoj crontab raspored te se taj raspored i akcije unutar njega pokreću pod tim korisnikom. Suština svrhe crontab-a je da dopušta korisniku da operacijski sustav izvršava određene skripte i naredbe bez njegove prisutnosti (tkz. schedule running).

Na slici broj devet se vidi sintaksa putem koje se crontab-u daje do znanja kada i što da izvede unutar operacijskog sustava. Zapis se sastoji od nekoliko zvjezdica koje označuju pojedini vremenski interval (minute, sati, dani u mjesecu itd.). Nakon odabranog intervala upisuje se korisnik koji izvodi naredbu i naposljetku naredba ili putanja do skripte. Primjer crontaba bi bio:

15 13 1 */3 * marko /home/marko/skripta.sh

U prijevodu bi ovo značilo da svaka tri mjeseca prvog dana u mjesecu u 13 sati i 15 minuta pod korisnikom „marko“ izvrši datoteku skripta.sh

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
```

Slika 9. Prikaz primjera rasporeda unutar crontab datoteke(*Crontab example,* 2014)

1.2.3. Oracle baza podataka

Po definiciji iz literature(Bryla B., 2015) baza podataka je skup međusobno povezanih podataka, pohranjenih u memoriji računala. Postoji u više oblika ali daleko najzastupljeniji je tkz. relacijski model koji je zasnovan na matematičkom pojmu relacije. Oracle je među prvima u industriji prepoznao potencijal relacijskog modela te su još davne 1983. godine izdali prve verzije relacijske baze podataka. Danas, Oracle je gigant koji predvodi industriju sa svojim novim produktima, među kojima je još uvijek baza podataka. Da bi naposljetku shvatili kako funkcionira naša procedura za oporavak moramo u nastavku pojasniti kako funkcionira jedna baza podataka ispod „haube“, odnosno objasniti ćemo arhitekturu baze podataka.

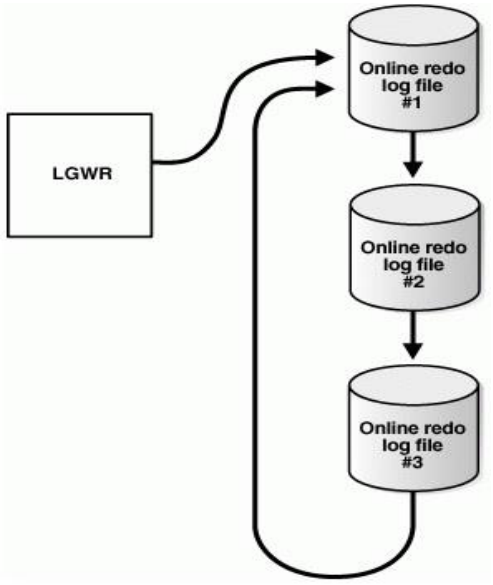
1.2.3.1. Arhitektura Oracle baze podataka

Moglo bi se reći da je baza podataka zbirka podataka koja se tretira kao jedinka. Svrha baze je pohraniti i dohvatiti povezane podatke. Poslužitelj (eng. DBMS) na kojem se baza nalazi je ključ za upravljanje informacijama. Općenito, poslužitelj pouzdano upravlja velikom količinom podataka u višekorisničkom okruženju tako da više korisnika može istodobno pristupati istim podacima. Također, poslužitelj se brine za sprječavanje neovlaštenog pristupa podacima i pruža učinkovita rješenja za oporavak od kvara. Oracle baza podataka sastoji se od najmanje jedne instance i jedne baze podataka. Instanca obavlja zadaće rukovanja memorijom i procesima, dok sa druge strane baza podataka se sastoji od skupa datoteka koja sadrže podatke. Samu arhitekturu možemo podijeliti na fizičku i logičku. Fizička bi predstavljala „stvarne“ datoteke koje postoje na tvrdim diskovima poslužitelja, dok bi logička predstavljala nešto što ne postoji u fizičkom obliku na tvrdom disku međutim definirano je od strane korisnika te kao takvo postoji unutar baze (tablice, indeksi, segmenti, tablična mjesta itd.). U nastavku ovog završnog rada fokusirati ćemo se samo na fizičku arhitekturu te objašnjenja koja nalazimo unutar literature (Rick Greenwald i dr., 2013) jer shvaćanje kako baza radi i zapisuje podatke je ključno kako bi mogli što bolje shvatiti sam proces procedure za oporavak. Dakle fizička arhitektura sastoji se od:

- Podatkovnih datoteka (eng. datafiles) – Svaka Oracle baza se sastoji od najmanje jednog fizičkog „datafile“-a koji sadrži sve podatke iz baze. Podaci koji su vezani za logičke komponente baze kao što su tablice i indeksi su fizički pohranjeni unutar podatkovnih datoteka. Podaci u njima se po potrebi čitaju tijekom normalnog rada baze podataka i pohranjuju u predmemoriju (eng. cache). Pohrana u predmemoriju vrši se iz razloga bržeg čitanja, u slučaju da korisnik zatraži neki podatak koji nije u predmemoriji onda se on učitava direkt iz podatkovne datoteke što može znatno duže potrajati. Također podaci koji su izmijenjeni ili na novo upisani unutar tablice se nužno ne zapisuju odmah unutar podatkovnih datoteka. Da bi se smanjila količina trenutaka pristupa disku i samim time povećale performanse, podaci se objedinjuju u memoriju i zapisuju u odgovarajuće podatkovne datoteke odjednom, kako je određeno postupkom pisanje baze podataka u pozadini (eng. DBW- database writer). Osim glavnih

podatkovnih datoteka bitno je istaknuti da postoji i „temp“ podatkovna datoteka, koji služi za pohranu privremenih podataka i unosa. Ona se automatski prazni po potrebi ili nakon ponovog pokretanja instance baze podataka.

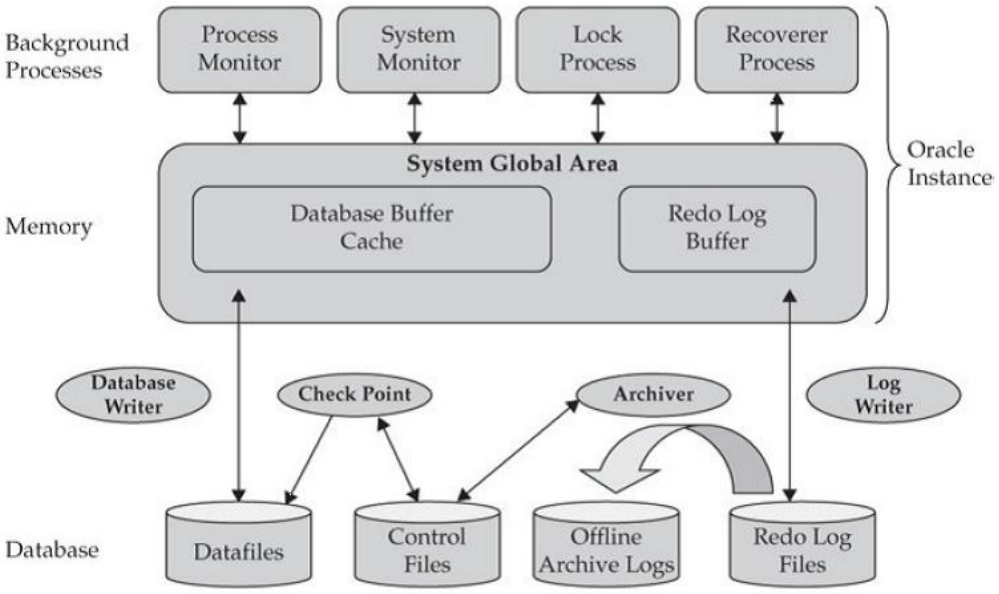
- Kontrolna datoteka (eng. control files) – Svaka Oracle baza podataka ima kontrolnu datoteku, kontrolna datoteka sastoji se od tekstualnih unosa koji određuju fizičku strukturu baze podataka. Neki od tih unosa uključuju naziv baze podataka, imena i fizičke putanje na disku do „redo“ datoteka i podatkovnih datoteka, vremenski žig izrade itd. Svaki put kad se instanca baze pokrene, prvo što radi je čitanje kontrolne datoteke putem koje identificira podatkovne datoteke, „redo“ datoteke i privremene podatkovne datoteke koje moraju postojati na tvrdom disku i biti spremne za rad kako bi baza mogla uspješno završiti postupak pokretanja. Također, bitno je naglasiti da se kontrolna datoteka koristi kod postupka oporavka baze podataka iz razloga što se u njoj nalazi i jedinstveni broj transakcije. Baza po tom broju zna što je sljedeći korak ka vraćanju na neko željeno stanje prije nego se dogodila neka greška. Više o tome ćemo vidjeti u nekom od sljedećih poglavlja ovog završnog rada kada budemo detaljno opisivali postupak oporavka.
- Datoteke dnevnika ponavljanja (eng. redo logs, online redo logs) – Svaka Oracle baza podataka sadrži skup od dvije ili više (najčešće tri) datoteke dnevnika ponavljanja. Te datoteke se mogu pojaviti u dva oblika, aktivne (eng. online) ili arhivske (eng. archive). Aktivne datoteke dnevnika ponavljanja sastoje se od „redo“ unosa putem kojih se bilježe sve promjene napravljene nad podacima unutar baze podataka. Baza time ostvaruje konzistentnost i sigurnost podataka jer ako dođe do greške sustava, sve napravljene promjene nad podacima mogu se aplicirati putem „redo“ datoteka te dobijemo natrag željeno stanje prije greške, napravljen rad i transakcije se nikad ne gube. Sama „redo“ datoteka je strukturirana tako da se sekvencijalno vrši zapis svih događaja nad podacima, ona po svojoj definiciji ima zadanu veličinu na disku te nakon što se popuni zapisivanje prelazi na sljedeću „redo“ datoteku po redu. Kad se sljedeća napuni prelazi na sljedeću i tako dalje. U trenutku kada je baza došla do zadnje „redo“ datoteke i nema više sljedeće vraća se na početnu, te postupak kreće iz početka usput gazeći stare zapise sa novima. Proces koji služi za zapisivanje unutar „redo“ datoteka nazive se LGWR (eng. log writer). Ovaj kompletni proces prikazan je na slici broj deset u nastavku.



Slika 10. Prikaz zapisivanja unutar datoteka dnevnika ponavljanja putem LGWR procesa(. *Reuse of Redo Log Files by LGWR,* 2008)

- Arhivirane datoteke dnevnika ponavljanja (eng. archive redo logs) – Kao što smo naveli baza u aktivne „redo“ datoteke zapisuju sekvencijalno. Kada se dođe do kraja „redo“ datoteka baza se vraća na početnu i kreće gaziti stare zapise o aktivnostima nad podacima sa novima. Samim time dolazi do gubitka prethodnih aktivnosti te u slučaju kvara više se ne mogu vratiti sve napravljene transakcije. U takvom slučaju eksplicitno se na razini cijele baze podešava tkz. „ARCHIVELOG MODE“ način rada. Takav način rada osigurava da u trenutku kada baza napuni jednu „redo“ datoteku te prijeđe ne drugu, ta napunjena datoteka se automatski kopira i šalje u arhivu definiranu od strane administratora sustava. Na taj način se osigurava postojanost svih zapisa o transakcijama koje su se dogodila unutar baze podataka neovisno o aktivnim „redo“ datotekama i LGWR procesu. Upravo će nam ovo biti bitno kod procedure za oporavak gdje ćemo koristiti arhivirane „redo“ datoteke kako bi držali „standby“ bazu ažuriranu sa najnovijim transakcijama iz originalne produkcijske baze.

Sve što smo prethodno naveli se također može vidjeti na prethodnoj slici broj jedanaest gdje možemo putem ilustriranih strelica zaključiti kako ide protok podataka unutar baze te koji proces radi koju zadaću i gdje upisuje informacije.



Slika 11. Prikaz arhitekture i procesa unutar Oracle baze podataka(*Oracle Database Internal Architecture*, bez dat.)

Također bitno je navesti kako sama instanca ali isto tako i baza mogu biti u više stanja, to je bitno radi činjenice da ćemo kod oporavka koristiti „Mount“ stanje baze u pripravnosti. Samim time trebamo i objasniti koja su sve to stanja i što znače, po definiciji literature (Darl Kuhn, 2013) baza se pojavljuje u:

- Ne montirano stanje (eng. nomount state) – Pokrećemo instancu međutim dajemo instrukciju da se ne montira baza podataka, ovakvo stanje nam služi kada želimo kreirati novu bazu podataka ili rekreirati kontrolnu datoteku iz već postojeće baze.

- Montirano stanje (eng. mount state) – Pokrećemo instancu i dajemo instrukciju da se montira baza podataka, u ovakvom stanju se odvija oporavak baze podataka. Baza u ovakvom stanju radi uredno, međutim raspoloživa je samo za čitanje od strane uskog broja korisnika i ne prima ostale konekcije i upite.

- Otvoreno stanje (eng. open state) – Dok je baza u montiranom stanju administrator može u bilo kojem trenutku dati instrukciju da prijeđe u otvoreno stanje. Otvoreno stanje je zapravo normalno stanje baze u kojemu je ona

raspoloživa za pisanje i čitanje te normalno prima konekcije i upite od strane korisnika.

Nakon što smo kroz nekoliko poglavlja objasnili sve bitne pojmove koji sudjeluju unutar naše procedure za oporavak i kad smo ustanovili što je to poslužitelj, mrežna oprema, virtualizacija operacijskog sustava, Solaris, bash ljuska i kad smo detaljno prikazali kako funkcionira baza podataka napokon možemo prijeći na suštinu ovog završnog rada, a to je procedura za oporavak baze podataka od nepredviđenih situacija.

2. PROCEDURA ZA OPORAVAK BAZE PODATAKA OD NEPREDVIĐENIH SITUACIJA

Prije nego što u nastavku prijeđemo na proceduru za oporavak istaknuti ćemo zašto je uopće potrebna. Dakle kao što smo naveli tvrtke i organizacije se u svim svojim djelatnostima oslanjaju na podatke, bilo da je riječ o industriji, turizmu ili nekoj drugoj grani gospodarstva. Kako vrijeme prolazi tako je i bitnost podataka sve veća, nekada su se koristili samo u svrhu dobivanja potrebnih informacija, danas postoje razne tehnologije koje omogućavaju operacije nad podacima koje nam daju puno više informacija nego nekad. Jedan od primjera bi bilo rudarenje koje sa današnjom tehnologijom omogućava da jedna tvrtka iz gomile sirovih podataka unutar baze podataka dobije razne analize na temelju kojih mijenja i utječe na svoje poslovanje. Svaka tvrtka radi ovoga mora uvidjeti važnost podataka te mora zaključiti da oni uvijek moraju biti osigurani. U tom trenutku većina njih se okrene klasičnoj sigurnosnoj kopiji (eng. backup) kao prvoj soluciji ka osiguravanju svojih podataka, upravo u nastavku ovog završnog rada ćemo vidjeti zašto to i nije tako kvalitetno rješenje.

2.1. Nedostaci klasičnog sigurnosnog kopiranja

Kao što smo prethodno naveli isprva kad je industrija zahtijevala da osigura svoje podatke počeli su razmatrati strategije i procedure sigurnosnog kopiranja podataka, te eventualnog obnavljanja u slučaju kvara ili gubitka podataka. Povoljni uređaji za sigurnosnu pohranu kao što su medijske trake se najčešće koriste , osim njih prakticira se i kopiranje sigurnosnih kopija na udaljena mjesta pohrane. Može se reći da je metoda sigurnosne kopije postala standardna praksa kod mnogih, međutim postoje i kod nje neki nedostaci. Pretpostavimo da se dogodi neki hardverski kvar ili katastrofa, da se trenutni produkcijski sustav na kojem obitavaju podaci unutar poslužiteljske sobe jedne tvrtke sruši. U tom trenutku ti podaci više nisu dostupni, niti neće biti dostupni sve dok se sigurnosna kopija ne obnovi na istom ili nekom drugom sustavu, ovisno o uvjetima. U slučaju da je primarna poslužiteljske soba nedostupna, prelazak na drugu i novu poslužiteljske sobu bio mi mukotrpan i jako vremenski zahtjevan. Upravo vrijeme oporavka je glavna briga i problem u takvim situacijama. Vrijeme povratka podataka će naravno ovisiti o količini podataka koje pokušavamo vratiti, međutim niti jedna tvrtka si ne može priuštiti zastoje u poslovanju radi vraćanja sigurnosne kopije. Uzmimo za primjer bilo koju međunarodnu banku, njihovo poslovanje si ne može priuštiti ni minutu zastoja dok traje proces obnavljanja, dakle treba im alternativno rješenje u slučaju kvara ili katastrofe. Osim samog vremenskog aspekta postoje i drugi nedostaci kod metode sigurnosne kopije, a to su:

- Kako se povećava količina podataka tako se i veličina sigurnosne kopije povećava. Samim time osim povećanog vremena obnavljanja, povećavaju se i troškovi držanja takve sigurnosne kopije
- Postoji mogućnost da se dogodi neki kvar ili zastoj kod procesa obnavljanja, što u konačnici rezultira činjenicom da se cijeli postupak mora ponovo ponavljati
- Postoje povećani troškovi hardvera kojim će se osigurati da performanse obnavljanja budu zadovoljavajuće
- Ako se sigurnosna kopija nalazi na udaljenom mjestu pohrane, treba uzeti u obzir i vrijeme potrebno da se ona kopira natrag na lokalnog poslužitelja, tek nakon potpunog kopiranja može započeti proces obnove

- Povremeno se sama sigurnosna kopija treba obnoviti i testirati, sam taj proces iziskuje troškove. Potreban je iz činjenice što jedino tako možemo ustanoviti je li naša sigurnosna kopija ispravna te da li možemo računati na nju u slučaju kvara ili gubitka podataka

Iz navedenih točaka može se zaključiti da niti jedna tvrtka ne bi trebala u potpunosti vjerovati i biti ovisna od sigurnosnoj kopiji u slučaju katastrofe. Upravo je to veoma istaknuto kod velikih baze podataka koje dosežu veličine do preko jednog terabajta podataka. Gdje bi vrijeme oporavka bilo veoma dugotrajno i poslovanja tvrtke bi bilo u stanju čekanja više desetaka sati. U takvim situacijama gdje je potrebno nešto više dolazi do implementiranja metode „Disaster recovery“ odnosno modela gdje je postoji egzaktna replika poslužiteljske sobe na udaljenoj lokaciji. Upravo naša procedura za oporavak baze podataka je dio takvog DR modela, te ćemo u nastavku ovog završnog rada proći kroz neke njegove prednosti.

1.2. Prednosti „Disaster recovery“ modela

Kao što smo već naveli tvrtke se ne mogu u potpunosti osloniti na sigurnosne kopije, te im trebaju neki drugi modaliteti zaštite od kvara ili katastrofe, jedan od tih modaliteta je i „Disaster recovery“. Dakle DR model podrazumijeva da postoji replika poslužiteljske sobe na udaljenoj lokaciji. Ta replika primarnog sustava trebala bi po hardverskim i softverskim svojstvima biti jednaka kao i originalna produkcijska okolina. Osim toga, procedurama za oporavak osigurati će se da se sve promjene na glavnom sistemu repliciraju na sekundarni sistem uz minimalna vremena kašnjenja. Najvažnije je istaknuti da u slučaju katastrofe ili kvara ne dolazi do potrebe za obnavljanjem iz sigurnosne kopije, već klijente unutar informacijskog sustava usmjerimo na sekundarni sustav i oni unutar minimalnog vremena nastavljaju sa svojim radom. Dakle, iz svega navedenog možemo uvidjeti da DR model ima puno prednosti, a brojni slučajevi iz prakse (Ravikumar i sur., 2019) istaknuli su neke od njih:

- Sekundarna okolina može se u bilo koje trenutku prebaciti i postati primarna okolina

- Tijekom razdoblja kada nema katastrofa ili kvarova, sekundarno okruženje može poslužiti za čitanje podataka. U slučaju baze podataka, baza u „recovery“ modu je raspoloživa za čitanje podataka, te može generirati izvještaje za korisnike i slično. Na taj način se smanjuje opterećenje primarne okoline.
- Osim za slučaj katastrofe sekundarna okolina nam može poslužiti za testiranje novih izmjena, pošto je po softveru i hardveru jednaka kao i primarna okolina. Implementacijom prvo na sekundarnu okolinu omogućuje se administratorima sustava da uvide i isprave sve potencijalne probleme prije nego tu istu izmjenu naprave na primarnom produkcijskom sustavu.
- U slučaju održavanje primarne okoline, gdje održavanje nalaže privremeno gašenje sustava. U takvim situacijama se može klijentsko okruženje prebaciti na sekundarni sustav, gdje nastavljaju normalno raditi do kraja održavanja primarnog sustava.

Iz svega navedenog mogli smo zaključiti da implementacija DR modela ima mnoge prednosti. Kao što je bilo spomenuto razvijene i implementirane procedure za oporavak se brinu da sekundarni sustav uvijek bude ažuran u odnosu na primarni sustav. U nastavku ovog završnog rada ćemo pobliže opisati jedan mali dio tih procedura, a to je procedura za oporavak baze podataka.

2.3. Postavljanje okoline

Radi potreba opisivanja i što boljeg shvaćanja procedure koju ćemo opisati u nastavku moramo postaviti neku zamišljenu informacijsko komunikacijsku okolinu. Unutar te okoline postoje korisnici u ulozi klijenata unutar informacijskog sustava. Korisnici od kojih se neki nalaze u Puli, dok se drugi nalaze u otprilike sto kilometara udaljenoj Rijeci. Kao što je prikazano na slici broj dvanaest naša zamišljena okolina se sastoji od dvije poslužiteljske sobe, od kojih je primarna soba u Puli dok je sekundarna u Rijeci. Unutar svake sobe postoji mrežna oprema koja osigurava međusobnu povezanost između dvaju poslužiteljske soba ali isto tako omogućuje klijentima pristup na poslužitelja i servise unutar njega. Kao što smo naveli postoji i poslužitelj, unutar kojeg se vrši više slojeva virtualizacije. Prvi sloj predstavlja VM model unutar kojeg se

virtualizira logička domena koja služi kao podloga u obliku operacijskog sustava za sljedeći sloj virtualizacije, a to je zona. Virtualna zona se putem OSV modela manifestira u obliku kontejnera te predstavlja zasebnu jedinku koja se vanjskom svijetu predstavlja kao zasebni operacijski sustav sa svojom mrežnom adresom. Unutar takve zone instalirana je Oracle-ova baza podataka, te ona služi kao glavno mjesto za pohranu svih podataka kojih koriste unutar naše zamišljene informacijske okoline.

Slika 12. Prikaz okoline kod procedure za oporavak baze podataka (autorski rad)

Konkretno u našem slučaju, radi lakšeg snalaženja u daljnjem opisu ćemo Pulsku virtualnu zonu nazvati „zonaPU“, a Riječku zonaRI. Dakle, zonaPU je produkcijska i primarna zona unutar koje se nalazi glavna baza podataka kojoj pristupaju svi korisnici informacijskog sustava, nebitno bili oni iz Pule ili iz Rijeke. Sa druge strane zonaRi je

zamišljena kao sekundarna zona te je na njoj instalirana također baza podataka ali je ona u stanju pripravnosti, ne radi ništa osim šta aplicira promjene koje nastaju na Pulskoj lokaciji uz lagano vremensko kašnjenje. Dakle proces oporavka nije zamišljen kao nešto što krene u datom trenutku kada je to potrebno te naposljetku završi, već je on stalan i u kontinuitetu se događa dok god korisnici vrše promjene nad primarnom Pulskom bazom podataka.

2.4. Tjedno kopiranje cijelog sistema

Svaki administrator koji upravlja nekim informacijskim sustavom teži ka tome da je sustav dostupan 100% vremena. Međutim u praksi je to otežano radi više faktora, jedan od tih faktora je i taj da procesi stvaranja sigurnosne kopije cijelog sistema puno bolje funkcioniraju u trenucima kada baza nije funkcionalna. Ako je baza ugašena u trenutku sigurnosne kopije cijelog sistema, to nam osigurava da je stanje uhvaćeno unutar TAR arhive sigurno konzistentno, jer u trenutku snimanja baza nije vršila nikakve transakcije te procesi unutar baze nisu vršili zapise po datotekama na disku. Da bi napravili sigurnosnu kopiju cijelog sistema prvo što moramo je dakle zaustaviti bazu podataka na zonaPU, to se u našem slučaju čini putem skripte unutar koje su zapisane ove bash naredbe:

```
###Spuštanje baze za tjedni full backup###
###Spusti listener###
lsnrctl stop
###spusti bazu###
sqlplus /nolog << EOF
connect / as sysdba
shutdown immediate;
exit;
EOF
```

Slika 13. Prikaz koda za gašenje baze podataka (autorski rad)

Kod komandnog jezika bash, znak „#“ označuje komentar, osim toga bitno je istaknuti da komanda „lsnrctl stop“ gasi „listener“ program odnosno alat koji sluša te prima konekcije prema bazi. Naredbom „sqlplus /nolog“ ulazimo u SQLPLUS komandu liniju,

gdje putem „EOF“ parametra dajemo da znanja Solarisu da u pozadini izvrši naredne linije naredbi. Dakle operacijski sustav se spaja na postojeću instancu baze u ulozi „sysdba“ što predstavlja najveće ovlasti unutar baze podataka i daje naredbu za gašenje. Nakon toga naša baza i operacijski sustav su spremni za stvaranje sigurnosne kopije. Na slici broj četrnaest vidimo jednostavnu skriptu koja se sastoji od jedne naredbe. Naredba govori sistemu da stvori arhivu na „backup“ putanji i da unutar nje uključi sve sa „/putanja_do_baze“. Usput da otvori tekstualnu datoteku i unutar nje da zapiše cjelokupni proces arhiviranja, na posljetku neka tu datoteku označi sa trenutnim datum i ekstenzijom „.log“.

```
#!/usr/bin/bash
####Stvori tjedni .tar backup svih putanja gdje je baza
tar -cvEf /backup_putanja/fullback.tar /putanja_do_baze > /logovi/fullback_`/usr/bin/date +%Y%m%d`.log
```

Slika 14. Prikaz koda za sigurnosnu kopiju cijelog sistema (autorski rad)

Nakon ovog završenog procesa, baza je ponovo spremna za podizanje, to činimo putem ovih naredbi:

```
##Dizanje baze nakon tjednog full backupa###
###Digni bazu###
sqlplus /nolog << EOF
connect / as sysdba
startup;
exit;
EOF
###Digni listener###
lsnrctl start
```

Slika 15. Prikaz koda za paljenje baze podataka (autorski rad)

Ovdje koristimo slične naredbe kao i kod procesa gašenja, međutim malo im je redoslijed okrenut. Skripta prvo ulazi unutar SQLPLUS komandne linije te nakon spajanja na instancu baze daje „startup“ naredbu koja govori bazi da se upali. Naposljetku ostaje samo startanje „listener“ procesa koji nakon pokretanja počinje sa slušanjem i primanjem konekcija.

Kao što smo u prethodnim poglavljima naveli da bi zakazali i rekli sistemu točno u kojem trenutku da pokrene skripte koje želimo koristili smo alat crontab čiju smo sintaksu već prethodno objasnili. Konkretno sami postupak gašenja baze, stvaranja sigurnosne kopije i ponovnog paljenja odvija se jednom tjedno najčešće u noćnim satima vikenda kada je najmanje aktivnosti na samom sistemu. Time dobivamo tjedno konzistentno stanje na kojega se možemo vratiti u bilo kojem trenutku ako je to potrebno. Samim time tu novonastalu arhivu šaljemo na sekundarnu lokaciju zonaRI pomoću naredbe „secure copy“. Na slici broj šesnaest vidimo da se skripta najprije pozicionira unutar mape gdje se nalaze sigurnosne kopije (eng. change directory naredba), zatim ih kopira na zonaRI. Alfanumerički zapis 172.xx.xx.xx predstavlja mrežnu adresu virtualne zone u Rijeci.

```
#!/bin/bash
## SCP fullbackupa za Disaster Recovery Rijeka
cd /fullbackup
scp fullback.tar root@172.xx.xx.xx:/fullbackup/
```

Slika 16. Prikaz koda za kopiranje sigurnosne kopije na lokaciju zonaRI (autorski rad)

Dakle sada imamo sigurnosnu kopiju baze iz Pule na zoni u Rijeci, međutim kao što smo naveli vrijeme obnove i povratka na željeno stanje iz sigurnosne kopije je jako dugotrajno, te ne želimo da nam jedini izvor povratka podataka i oporavka od greške bude arhivska datoteka. Upravo radi toga uspostavljamo tkz. „log shipping“, više o tome u sljedećem poglavlju ovog završnog rada.

2.5. Slanje i apliciranje arhivskih REDO datoteka

„Log shipping“ metoda, odnosno stanje kod kojega baza na zonaRI prima promjene od baze sa zonaPU unutar kratkih vremenskih intervala te primjenjuje nastale promjene, na taj način je uvijek ažurna i spremna preuzeti ulogu primarne baze podataka. To se postiže na način da sekundarnu bazu postavimo u „mount“ stanje,

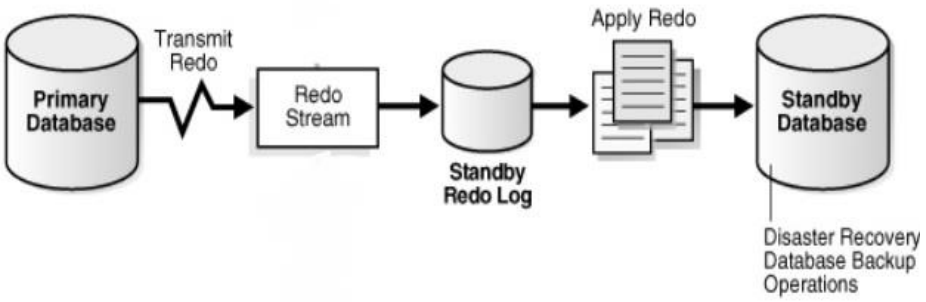
koje smo prethodno unutar ovog rada i objasnili. U montiranom stanju baza je raspoloživa samo za čitanje i obnovu (eng. recovery).

Sa druge strane, na zonaPU baza radi normalno, u sklopu tog normalnog rada piše REDO datoteke i prebacuje napunjene datoteke u arhivu. Upravo te arhivske REDO datoteke sadrže sve promjene i transakcije koje su nastale unutar primarne baze, te ćemo pomoću njih te iste promjene primijeniti na Riječkoj lokaciji. Dakle, prvi korak nam je automatizirati slanje prema zonaRI svih arhiviranih REDO datoteka, to činimo pomoću naredbe RSYNC. Na slici broj sedamnaest možemo vidjeti proces kopiranja svih arhiviranih redo datoteka sa lokacije „/arch“ unutar zonaPU prema zonaRI.

```
#!/usr/bin/bash
# prepis redo logova svakih 5 minuta prema RI baza ----------
rsync -avH /arch/* oracle@172.xx.xx.xx:/arch/. >> /skripte/redo_rsync.log
```

Slika 17. Prikaz koda za automatsko kopiranje arhiviranih redo datoteka iz Pule prema Rijeci (autorski rad)

Skripta je namještena tako da se je unutar crontab-a specificiralo da se pokreće svakih pet minuta, nakon pokretanja gleda sve što postoji na „/arch“, a nedostaje na istoj lokaciji unutar zonaRI te to kopira, odnosno radi tkz. delta kopiranje kao što smo prethodno naveli. Veoma je bitno naglasiti da se kod ovakvih procedura uvijek uzimaju arhivske redo datoteke koje su nastale nakon cjelokupne sigurnosne kopije sistema, iz razloga što jedino tako možemo osigurati konzistentnost. U kratko, uzme se kopija sistema iz Pule, kopira se i raspakira u Rijeci, te se na tu kopiju počinju aplicirati arhivske redo datoteke. Na taj način smo sasvim sigurni da transakcije kreću od onog trenutka od kojeg trebaju. Osim nas zato se brine i kontrolna datoteka unutar baze podataka, koja pomoću točnog internog broja transakcije zna koju arhivsku datoteku treba aplicirati sljedeću, te neće nastaviti apliciranje ako fali jedna datoteka u cjelokupnom lancu. Cijeli taj proces ćemo pobliže vidjeti u nastavku. Sve što smo prethodno naveli prikazano je na slici broj osamnaest, gdje se jasno vidi cjelokupni životni vijek jedne redo arhivske datoteke.



Slika 18. Prikaz procesa kopiranja i apliciranja redo datoteka (Ravikumar i sur., 2019)

Nastaje na primarnoj bazi, zatim se kopira na sekundarnu lokaciju te tamo se aplicira i prenosi sve promjene sa primarne na sekundarnu. U našem slučaju samo apliciranje na sekundarnoj bazi se vrši pomoću naredbi na slici broj devetnaest.

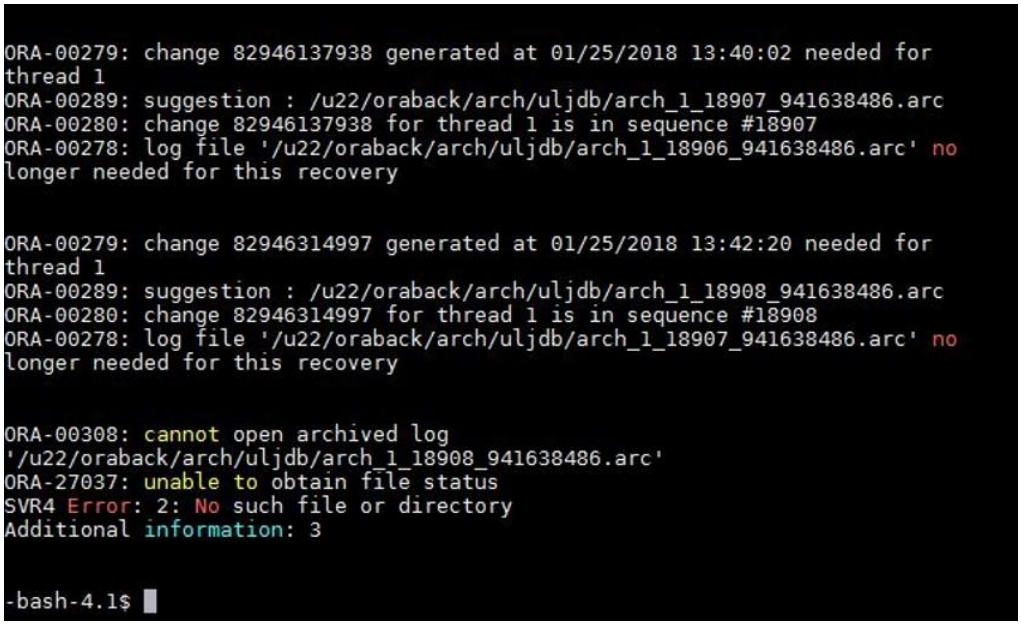
Samo proces pokreće se ručno, jer radi kontinuirano. Na slici u nastavku su prikazane već sve standardne i spomenute komande. Jedino valja istaknuti „RECOVER database using backup controlfile until cancel“ koja govori bazi da pogleda unutar kontrolne datoteke i da ustanovi koji joj po redu dolazi sljedeći broj transakcije. Nakon toga neka provjeri postoji li unutar definiranog mjesta za arhivske redo datoteke, u našem slučaju „/arch“ neka redo datoteka koja odgovara broju transakcije koja slijedi, ako postoji neka ga aplicira. Na taj način baza kontinuirano provjerava i aplicira sve nove arhivske redo datoteke koji pristižu iz Pule.

```
#!/bin/bash
# Apliciranje arch logova na Disaster Recovery Rijeka
$ORACLE_HOME/bin/sqlplus -s /nolog << eof
connect / as sysdba;
RECOVER database using backup controlfile until cancel;
auto
exit;
eof
```

Slika 19. Prikaz koda za apliciranje arhivskih redo datoteka (autorski rad)

Sa slike broj dvadeset je bitno istaknuti da je ovaj „change number“ zapravo taj broj transakcije kojeg smo spominjali. Pri kraju slike također vidimo da je baza odgovorila da nema sljedeće redo datoteke u lancu, to je zato što je svjesna činjenica koja bi trebala ići sljedeća, međutim ta sljedeća u trenutku kada je nastala slika nije još stigla

sa primarne baze u Puli, čim se pojavi baza će ju aplicirati. Na taj način se održava konzistentnost i ažurnost baze uz minimalno kašnjenje za originalnom Pulskom bazom podataka.



Slika 20. Prikaz apliciranja arhivskih redo datoteka na sekundarnoj bazi (autorski rad)

U slučaju da se dogodi neka greška sistema u Puli, ili bilo kakva prirodna katastrofa koja bi sasvim onesposobila primarnu poslužiteljske sobu, sekundarna poslužiteljske soba u Rijeci je spremna prihvatiti svo opterećenje sustava i upite korisnika. Dovoljno nam je u određenom trenutku samo prebaciti bazu u otvoreno stanje i uputiti sve poslovne aplikacije na drugu bazu podataka. Na taj način smo se zapravo oporavili od nepredviđene situaciju u jako malom vremenu. Ne oviseći o sigurnosnim kopijama i raznim drugim alatima.

2.6. Učestalost i prihvaćenost u praksi

Na prvi pogled i nakon što smo istaknuli sve mane i nedostatke klasičnog oporavka putem sigurnosne kopije mogli bi smo zaključiti da oporavljanje sistema putem procedure za oporavak i sekundarne udaljene okoline uopće nema nedostataka,

međutim u praksi su stvari malo drugačije. Prva stvar koju možemo izdvojiti kao nešto što zasigurno treba istaknuti je kompleksnost. Kroz prethodna poglavlja ovog završnog rada vidjeli smo kako funkcionira naša procedura za oporavak te uvidjeli koliko malih skripta odnosno komadića koda radi zajedno unutar jednog vremenskog intervala da se uspješno obavi apliciranje promjena sa Pulske baze podataka na Riječku. Kompleksnost sama po sebi ne mora predstavljati nedostatak, međutim u realnoj situaciji gdje kontinuitet poslovanja jedne tvrtke ovisi u ažurnosti sekundarne baze podataka , bi zahtijevala visoko obučeno osoblje u ulozi administratora sustava kako bi se osiguralo da ta navedena procedura zasigurno uvijek radi. Visoko stručno osoblje iziskuje i veće financijske troškove, osim troškova za njih postoji i fiksni trošak preslike poslužiteljske sobe. Naime kao što smo već prethodno spomenuli sekundarna poslužiteljske soba na lokaciji u Rijeci bi trebala po hardverskim svojstvima biti jednaka kao i ona primarna u Puli. Što zapravo jednoj tvrtki predstavlja duple troškove za informacijsko komunikacijsku opremu, jer što se kupi u Puli, se mora ponovo kupiti i za u Rijeku.

Tu upravo dolazi do vječnog pitanja da li se isplati uložiti u bolji sustav oporavka ili riskirati da se poslovanje zaustavi na nekoliko sati dok traje procedura oporavka putem sigurnosne kopije te za to vrijeme izgubiti novac.

Ako našu zamišljenu okolinu postavimo u neke realne gabarite, npr. unutar neke industrije koja se bazira na proizvodnji na traci, dakle proizvodni proces unutar kojega se prate jasno definirani vremenski rokovi te je jednostavno nedopustivo da traka iz razloga nekakvog kvara stane. Također zamislimo da u pozadini cijeli proizvodni proces se bazira da na bazi podataka od otprilike jednog terabajta podataka, u slučaju da ne radi baza proizvodnja staje. Postoji puno mogućih loših ishoda i situacija unutar kojih nešto može proći po krivu, međutim konkretno u našem slučaju promatrati ćemo samo dva. Slučaj u kojemu dolazi do gubitka podataka te se dešava tkz. softverski kvar na sloju same baze podataka i slučaj u kojemu dolazi do teškog hardverskog kvara uzrokovanog prirodnom katastrofom ili nekim lošim atmosferskim uvjetima (poplava, potres itd.). U prvom slučaju nakon kvara postoji mogućnost da obnovimo bazu podataka iz sigurnosne kopije, samo po sebi obnavljanje traje nekoliko sati, te tih nekoliko sati proizvodnja čeka. Financijski neisplativo, međutim u pogledu puno tvrtki i dalje podnošljivo. Jer manje novaca će se potencijalno izgubiti unutar tih par sati čekanja nego što će trebati ulaganja u potpunu sekundarnu okolinu za oporavak. No,

šta ako se dogodi drugi slučaj, što ako primarna poslužiteljske soba u potpunosti nastrada u nekakvoj poplavi i potresu? Tada obnova iz sigurnosne kopije ne dolazi u obzir jer ne postoji okolina gdje će se podaci vratiti te može se vrlo jasno zaključiti da ako bilo koja tvrtka čiji je proizvodni proces ovisan o informacijskom sustavu dočeka takvu situaciju ne pripremljena to može označiti financijski krah i kraj takve tvrtke. Upravo radi ovakvih potencijalnih situacija se isplati uložiti u sekundarnu okolinu za oporavak, te samim time postaviti kontinuitet poslovanja na prvo mjesto.

3. ZAKLJUČAK

Kroz ovaj završni rad vidjeli smo što je jedna jednostavna informacijsko komunikacijska okolina unutar koje obitava jedna baza podataka, upoznali smo virtualizaciju putem VM for SPARC tehnologije i sve slojeve od kojih se ona sastoji. Također smo saznali što je to korisnička ljuska BASH i koji su sve to alati i naredbe koje koristimo kod dnevne administracije i uporabe poslužitelja baze podataka. Bitno je istaknuti da smo jasno objasnili što stoji iza jedne relacijske baze, što su to podatkovne podatke, a što REDO datoteke te kako funkcioniraju stvari ispod „haube“, što je zapravo bila i suština shvaćanja naše procedura za oporavak od nepredviđenih situacija. Osim toga, shvatili smo zašto oslanjanje na samo sigurnosnu kopiju podataka nije dobra praksa te uvidjeli koje su prednosti DR modela.

Naposljetku smo objasnili našu proceduru za oporavak baze podataka tako da smo postavili zamišljenu okolinu koja se sastojala od dvije poslužiteljske sobe na udaljenim lokacija, putem skriptiranja i naredbi prikazali smo repliciranje jedne poslužiteljske sobe na drugu. Samim time omogućili smo oporavak od nepredviđenih situacija te smo predstavili alternativu za klasično vraćanje iz sigurnosne kopije.

Postavljanjem naše procedure u realne gabarite također smo istaknuli da osim svih dotada navedenih prednosti kao što su brzina oporavka i fleksibilnost postojanja rezervnog sekundarnog sustava postoje i neki nedostaci. Nedostaci koji se pojavljuju u obliku kompleksnosti koja za sobom vuče financijske troškove. Da li oni bili za osoblje zaduženo za održavanje takvog sustava i njegove replicirane kopije ili za opremu koju kupujemo duplo kako bi postigli maksimalnu usklađenost dvaju sustava i izbjegli potencijalne greške kod replikacije koje bi se mogle pojaviti radi eventualnih razlika u hardveru i načinu na koji je on postavljen. Unatoč svemu tome bitno je shvatiti kako je od krucijalne važnosti svakog poduzeća financijski boljitak i isplativost, te imajući to u vidu na prvi pogled se troškovi sekundarne replicirane okoline čine bespotrebni i nevažni, međutim daljnjim promišljanjem zapravo se dolazi do drugačijeg zaključka. Primarni je cilj svake tvrtke da sačuva kontinuitet svog poslovanja, a u suštini tog poslovanja se nalazi podatak odnosno informacija unutar njega. Samim time se

svakako isplati implementirati nekakvu proceduru za oporavak od nepredviđenih situacija, pa makar nikad to takve situacija niti ne došlo.

POPIS LITERATURE

1. Greenwald R., Stackowiak R., Stern J.(2013). *Oracle Essentials, Oracle Database 12c,* peto izdanje, Sebastopol, O'Reilly Media

2. Victor J., Savit J., Combs G., Netherton B. (2017). *Oracle Solaris 11 System Virtualization Essentials*, Redwood Shores, Prentice Hall

3. Kuhn D. (2013). *Pro Oracle Database 12c Administration*, Apress,

4. Bryla B (2015). *Oracle Database 12c DBA Handbook*, Oracle Press

5. Calkins B. (2013). *Oracle Solaris 11 System Administration*, Prentice Hall

6. Ravikumar Y.V., Basha N., Krishna Kumar K.M., Mukund Sharma B., Kerekovski K. (2019). *Oracle High Availability, Disaster Recovery, and Cloud Services*, Apress

7. Albing C., Vossen J.P., Newham C. (2007). *Bash cookbook*, prvo izdanje, Sebastopol, O'Reilly Media

8. Robbins A., Beebe N.H.F. (2005). *Classic shell scripting*, prvo izdanje, Sebastopol O'Reilly Media

9. Robbins A. (2006). *Unix in a nutshell*, prvo izdanje, Sebastopol, O'Reilly Media

10. Žagar M. (1995). *Unix i kako ga koristiti*, Sveučilište u Zagrebu, Fakultet elektrotehnike i računalstva

11. *TAR manual pages* (2020). preuzeto sa https://man7.org/linux/man-pages/man1/tar.1.html

12. *RSYNC manual pages* (2020). preuzeto sa https://man7.org/linux/man-pages/man1/rsync.1.html

13. *Server room example* [Slika] (2017). preuzeto 25.08.2020 sa https://www.anexio.com/keep-server-room-tip-top-condition/

14. *Cisco ethernet switch* [Slika] (bez dat.) preuzeto 25.08.2020 sa https://www.m4l.com/SRW224G4P-K9-EU-Cisco-Switches

15. *LAN and WAN* [Slika] (bez dat.) preuzeto 26.08.2020 sa https://sites.google.com/site/wikiexampleitsn/home/lan-and-wan

16. *Oracle SPARC T5-2 Server* [Slika] (bez dat.) preuzeto 26.08.2020 sa https://www.bsi.uk.com/sparc-t5-2-server

17. *Zone Clusters Whitepaper* [Slika] (2011). preuzeto 2.09.2020 sa https://www.oracle.com/technetwork/server-storage/solaris-cluster/documentation/zone-clusters-why-whitepaper-322070.pdf

18. *Crontab example* [Slika] (2014). preuzeto 16.09.2020 sa https://stackoverflow.com/questions/26472855/crontab-on-centos-6-5-not-working

19. *Reuse of Redo Log Files by LGWR* [Slika] (2008). preuzeto 16.09.2020 sa http://docs.oracle.com/cd/B28359_01/server.111/b28310/onlineredo001.htm

20. *Oracle Database Internal Architecture* [Slika] (bez dat.) preuzeto 16.09.2020 sa https://www.oreilly.com/library/view/sap-basis-administration/9780071663489/ch19lev1sec1.html

POPIS SLIKA

SAŽETAK

U sklopu ovog završnog rada cilj nam je bio razvoj jedne procedure za oporavak baze podataka od nepredviđenih situacija. Kroz uvodna poglavalja objasnili smo što je to informacijski sustav, to koje su to njegove komponente ključne za shvatiti i poimati prije nego se objasni kako djeluje procedura za oporavak. Vidjeli smo što je to poslužitelj, mrežna oprema, virtualizacija, što su logičke domene, a što virtualne zone. Isto tako objasnili smo što je operacijski sustav Solaris te koje alate unutar ljuske BASH koristimo kod navedene procedure. Osim toga, bitno je bilo i shvatiti kako jedna baza podataka radi i djeluje ispod površine, na sistemskoj razini. Naposljetku smo sve pokazano i spomenuto postavili unutar jedne zamišljene okoline gdje smo putem skriptiranja replicirali primarnu bazu podataka na sekundarnu uz minimalna vremena kašnjenja. Na taj način stvorili smo sekundarnu okolinu koja je spremna za preuzeti teret primarne u slučaju greške ili kvara.

Ključne riječi: Oracle baza podataka, sekundarna okolina za oporavak, Solaris, UNIX, bash skriptiranje

ABSTRACT

As part of this bachelor’s thesis, our goal was to develop a procedure for recovering the database from unforeseen situations. Through the introductory chapters, we explained what an information system is, what its components are key to understanding and comprehending before explaining how the recovery procedure works. We have seen what a server is, network equipment, virtualization, what are logical domains, and what are virtual zones. We also explained what the Solaris operating system is and what tools we use inside the BASH shell for this procedure to work. In addition, it was important to understand how a database works and operates below the surface, at the system level. In the end, we set up everything shown and mentioned within one imaginary environment where we replicated the primary database to the secondary one with minimal delays through scripting. Thus creating a disaster recovery site.

Key words: Oracle database, disaster recovery site, Solaris, UNIX, bash scripting