Univerzitet u Beogradu
Matematički fakultet

Migracija između RSUBP Orakl i DB2
-master rad-

Student: Miljan Danilović Mentor: dr. Nenad Mitić

Beograd, Oktobar 2012 god.

Sadržaj

1. Uvod

Migracija baze između različitih RSUBP-ova je složen proces, sastavljen od više aktivnosti, kojim se iz starog informacionog sistema baza sa pripadajućim objektima, aplikacijama i podacima prenosi u novi sistem, pri čemu se vrše sve potrebne izmene nad objektima, aplikacijama i podacima prema zahtevima novog sistema.

Migraciju često otežavaju činjenice kao što su potreba za menjanjem arhitekture i dizajna izvorne baze kako bi se izašlo u susret novim funkcionalnostima u novom okruženju, potreba za optimizacijom i racionalizacijom implementirane poslovne logike prilikom prelaska na novi sistem, zahtevi za implementacijom novih poslovnih pravila, više različitih lokacija i različitih vrsta izvora iz kojih podatke treba migrirati u bazu na novom sistemu. U životnom veku starog sistema uglavnom je bilo više dogradnji i primene različitih informacionih tehnologija što za posledicu ima stanje u kome objekti i podaci u bazi nisu organizovani na jedinstveno uređen način.

Proces konverzije sačuvanih objekata u bazi i aplikacija koje rade nad bazom kao i prenosa podataka nije jednostavan ni u slučajevima kada se vrši migracija samo jedne baze podataka starog sistema u bazu novog sistema zbog toga što je dizajn baze novog RSUBP-a uvek bitno različit od dizajna baze starog sistema. Poredeći sadržaj starog i novog informacionog sistema generalno su nad kolonama u tabelama definisani tipovi koji su svojstveni za izvorni sistem, da objekti baze kao što su procedure, funkcije i okidači napisani u proceduralno nadograđenom SQL programskom jeziku koji je takođe svojstven za stari sistem, da su podaci često različitih tipova, smešteni u različite tabele i u različitim formatima. Često postoji potreba da objekti i podaci u novom sistemu budu međusobno povezani na drugačiji način nego u starom zbog primene različitih poslovnih pravila i različitih aplikativnih rešenja. Čak i kod veoma uređenih izvornih baza migracija nije jednostavna, jer se može raditi o više stotina tabela i više miliona zapisa. Zbog svega ovoga migracija baze je mnogo širi pojam nego što je samo jednostavno prepisivanje definicija objekata, koda aplikacija koje rade nad bazom i prenos podataka na novi sistem. Ona obuhvata sve aktivnosti potrebne da se iz starog informacionog sistema objekti, aplikacije i podaci prikupe, izvrše potrebne izmene I transformacije nad njima, prenesu u novi sistem i izvrši provera ispravnosti konverzije i prenosa.

Razloga za zamenu postojećeg relacionog sistema novim, a samim tim i za migraciju objekata, aplikacija i podataka baze ima više, medju kojima je najčešći modernizacija. Kompanije rastu, menjaju se i usvajaju nove biznis strategije. Troškovi održavanja zastarelih informacionih sistema, kao i razvoj zastarelog aplikativnog softvera mogu da budu veoma značajna stavka u IT budžetu. Visok, skoro eksponencijalni porast obima podataka koje informacioni sistem obrađuje i čuva zahteva stalnu modernizaciju. Migracija postojećih objekata, aplikacija i podataka na moderan, stabilan i snažan sistem za upravljanje bazom podataka postaje neophodan uslov da bi kompanija mogla da prati brze promene poslovnog okruženja.

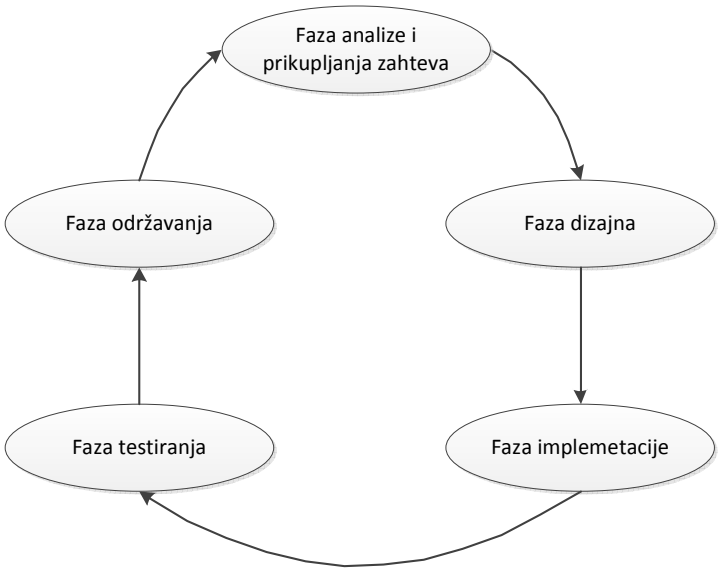
U okviru ovog master rada prikazana je migracija između relacionih sistema za upravljanje bazom podataka Orakl i DB2. Najpre je dat prikaz opšte metodologije projekta migracije koji je primenljiv na opisane slučajeve, potom je opisan scenario migracije DB2 baze na Orakl RSUBP i u poslednjem delu je opisana migracija u suprotnom smeru, odnosno migracija Orakl baze na DB2 RSUBP. Prilikom opisa samog procesa migracije dat je prikaz svih koraka koje je neophodno odraditi kako bi se uspešno izvršila migracija na novi sistem.

2. Metodologija projekta migracije

Kao i svi drugi IT projekti i projekat migracije zahteva pažljivo isplaniranu i dobro definisanu metodologiju koja će obezbediti uspešno izvršavanje procesa migracije. Svrha metodologije za migraciju je da pruži opšti okvir sa unapred definisanim i upravljivim aktivnostima koje rezultiraju uspešnim i efikasnim procesom migracije baze podataka sa jednog RSUBP-a na drugi bez obzira na njihovu unutrašnju organizaciju ili platformu na kojoj su postavljeni. Kada se vrši migracija baze sa jedne platforme na drugu onda je svakako od interesa da se što veći deo dizajna izvorne baze, posebno to važi za dizajn šeme, prenese na ciljnu platformu. Često je potrebno izvršiti određene izmene na dizajnu i arhitekturi baze kako bi bilo moguće iskoristiti sve nove mogućnosti koje pruža novi sistem koje ujedno i najčešće predstavljaju razlog migracije. Da bi se postigao određeni nivo racionalizacije i što bolja optimizacija tokom procesa migracije potrebno je što bolje shvatiti životni ciklus jednog projekta migracije i važnost svake faze u njemu. Mnogi faktori mogu uticati na uspeh jednog projekta migracije, kao što je raspoloživost kvalifikovanog IT osoblja, alata i tehnologija za migraciju, razumevanje izvorne i ciljne arhitekture baze i realno planiranje projekta. Identifikovanje ovih faktora pre početka jednog projekta migracije može ubrzati izvršavanje procesa i pomoći veštom rešavanju svih problema i izazova koji se mogu naći na putu do završetka projekta.

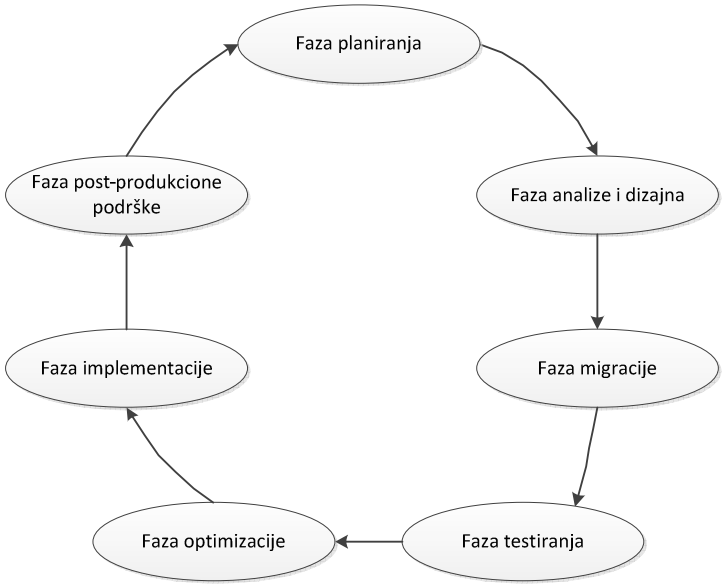
2.1 Metodologija vodopada

Da bi se smanjili rizici i dobio što pozitivniji efekat metodologija mora biti definisana pre samog početka projekta. Tradicionalna metodologija vodopada se sastoji od sledećih razdvojenih faza: prikupljanje zahteva, faze dizajna, implementacije, testiranja i na kraju faze održavanja. Ona može biti predstavljena sledećim dijagramom:



Metodologija vodopada može biti primenjena na projekat migracije, s tim što se faza analize i prikupljanja zahteva često zamenjuju fazom planiranja. Na primer, za migraciju u toku koje dolazi do potpune ili značajne izmene postojećih aplikacija faza prikupljanja i analize zahteva je primenljiva. U

velikom broju slučajeva migracija baza sa jedne platforme na drugu, ne postoji potreba za ponovnom analizom i prikupljanjem zahteva jer postojeće aplikacije pokrivaju sva poslovna pravila i novi zahtevi ne postoje. Tako da se najčešće umesto faze analize i prikupljanja podataka u tim slučajevima koristi faza planiranja u okviru koje se upoznaje trenutno okruženje baze i određuje najbolji način da se baza migrira u ciljno okruženje. Za razliku od metodologije vodopada u kojoj se faze izvršavaju jedna za drugom, moguć je scenario u kome se faze u toku životnog ciklusa projekta migracije izvršavaju paralelno. Tako na primer testiranje funkcionalnosti i performansi određenih komponenti može biti izvršeno a da čitava faza migracije i nije dovršena. Migracija baze sa jedne platforme na drugu može zahtevati određene izmene koje su prouzrokovane određenim tehničkim razlikama između različitih platformi. Sledeći dijagram ilustruje najčešći životni ciklus jednog projekta migracije.



2.2 Faza planiranja

Faza planiranja u projektu migracije je slična fazi prikupljanja i analize zahteva u metodologiji vodopada za razvoj softvera. Umesto prikupljanja zahteva posmatrano iz perspektive jednog projekta razvoja softvera u ovoj fazi se skupljaju informacije koje olakšavaju upravljanje projektom, ukazuju na troškove prilikom migracije, ukazuju na različite pristupe pri migraciji, koji alati bi se mogli koristiti itd. U okviru ove faze je potrebno napraviti potpunu sliku o svim komponentama aplikacija koje se migriraju kako bi se oformila procena uticaja zamene platforme na kojoj radi aplikacija u smislu njene interakcije sa servisima za integraciju, izveštavanje, procese za “bekap” i oporavak. Da bi faza planiranja bila iscrpna potrebno je razmotriti određene teme od interesa za ovu fazu:

Potreba za migracijom – Veoma je važno razumeti razloge migracije. Na primer, ako licenca za legalnost softvera na kome se trenutno nalazi baza uskoro ističe, potreba za migracijom na drugi sistem je urgentna. Takođe, produžavanje ugovora podrške za sistem na kome je baza veoma je skup i nije isplativ za klijenta tako da ukoliko ne postoji potreba za nekim većim izmenama na postojećim aplikacijama proces migracije baze na novi RSUBP se može sprovesti bez velikog napora.

Poznavanje okruženja – Za svaku aplikaciju je potrebno napraviti detaljan spisak komponenti sa kojima intereaguje u svom radu. Ovaj pristup pomaže u određivanju obima i količine napora potrebnog da se proces migracije uspešno privede kraju. To uključuje obuhvatanje svih informacija koje se odnose na broj programa, skriptova, eksternih interfejsa, verziju operativnog sistema, verziju baze i slične informacije.

Alati za migraciju - Često se javlja potreba za testiranjem različitih alata za migraciju kako bi se izvršila procena njihove efikasnosti i preciznosti. Visok nivo automatskog obavljanja poslova sa dozvoljenom preciznošću rezultuje sa manjim utroškom vremena u fazi migracije i testiranja.

Pružaoci usluga migracije – Klijenti se često odlučuju na podršku od strane pružalaca usluga poslova migracije jer nemaju dovoljno obučeno IT osoblje koje bi moglo uspešno izvršiti projekat migracije. Najčešće pružaoci podrške za migraciju koriste svoje alate za planiranje i migraciju.

Procena napora uloženog u migraciju – Ovu procenu najčešće vrši pružalac podrške pri migraciji ili vendor sistema za upravljanje bazom podataka i ona predstavlja najčešći poslovni zahtev pred početak jednog projekta migracije. Procena uloženog napora za uspešan završetak projekta migracije zavisi od mnogo faktora u koje spadaju veličina baze podataka i njenih aplikacija, komponenti itd.

Trening i obuka IT osoblja – Klijent najčešće zahteva od pružaoca podrške obuku za svoje IT osoblje. Ona podrazumeva upoznavanje IT osoblja sa novom platformom na koju se vrši migracija. Očekuje se da pošto prođe obuku IT osoblje bude osposobljeno za rad na novoj platformi i da učestvuje u samom procesu migracije ukoliko je to potrebno. Dakle, potrebno je definisati odgovarajuće programe za trening IT osoblja koji će biti bazirani na ulogama osoblja u poslovnoj organizaciji. Takođe je potrebno preneti znanje od strane tima za migraciju na timove za razvoj i administraciju koji će u budućnosti raditi sa bazom na novoj platformi.

IT resursi potrebni za ciljni sistem - Zahtevi za razvoj novog okruženja takođe treba da budu dobro isplanirani. Planiranje u ovom slučaju mora uzeti u obzir kritične komponente, funkcionalnosti baze kao i dodatni softver. Oni mogu biti potrebni za podršku pri migraciji ili za održavanje posle uspešno izvršene migracije. Ovi resursi najčešće podrazumevaju hardverske i softverske komponente.

IT resursi potrebni za projekat migracije – Softverski i hardverski resursi potrebni za poslove u sklopu projekta migracije takođe moraju biti identifikovani pre samog početka projekta. Najčešće klijenti moraju kupiti dodatni softver a ponekad i hardver za poterbe projekta migracije. U skorije vreme ove usluge se mogu dobiti i od strane pružaoca usluga u vidu hardverskih i softverskih resursa putem tehnologije oblaka.

U jednom projektu migracije uvek treba odvojiti dovoljno vremena za fazu planiranja kako bi se kao rezultat te faze dobila potpuna i sadržajna procena poslova migracije. Kada se primenjuje detaljno i iscrpno planiranje u sklopu projekta migracije, najčešće se koristi niz softverskih alata koji daju celopkupan snimak svih potrebnih informacija iz izvornog sistema. Ove informacije pomažu prilikom analize zavisnosti između aplikacija, komponenti baze podataka, kao i količine truda koji je potrebno uložiti u samom procesu. Ovi alati analiziraju svaki program i svaku liniju koda u njima kako bi se dobila jasnija slika o njihovoj složenosti. Informacije koje nam oni pružaju mogu pomoći pri proceni uticaja procesa migracije baze na već prethodno razvijene aplikacije koje pokrivaju poslovne procese organizacije. U te aplikacije spadaju:

Programi koji direktno intereaguju sa bazom – Postoje alati koji mogu direktno da pronađu broj aplikacija koje zahtevaju intervencije na ugnježdenim SQL upitima u njihovom kodu ili intervencije koje nastaju kao posledica specifičnog API-ja nove baze.

Programi koji izvršavaju direktno transakcije nad bazom – Identifikovanje ovih programa je od posebnog značaja jer na njihov rad može negativno uticati transakciono ponašanje nove baze. U njih sapadaju sledeće vrste programa:

- Programi koji imaju eksplicitne kontrolne naredbe kao što su COMMIT ili ROLLBACK. Ovi programi drže direktnu kontrolu nad transakcijama koje su sami otpočeli.
- Programi koji nemaju direktnu kontrolu nad transakcijom ali pozivaju procedura koje otpočinju transakciju. U ovom slučaju procedura ima direktnu kontrolu nad transakcijom.
- Programi koji u svom kodu sadrže DML naredbe ali nemaju kontrolu nad transakcijom. U mnogim slučajevima njima upravljaju drugi programi koji na taj način izvršavaju transakciju i dobijaju potrebni povratni rezultat.

Programi ili skriptovi za rad sa podacima iz baze – Ovi programi takođe moraju biti izmenjeni zbog specifičnosti koji se odnose na alate baze koje koriste za poslove rada sa podacima. Takođe se mora obratiti posebna pažnja na specifične komande, ugnježdeni SQL i korišćene parametre, koji mogu biti nekompatibilni u novom okruženju baze.

Identifikovanje broja skriptova koji se koriste za poslove upravljanja i administracije baze pomaže prilikom pravljenja procene potrebnog vremena i truda za njihovo jednostavno prepisivanje ili potpuno menjaje kako bi se prilagodili specifičnim alatima za praćenje i administraciju novog RSUBP-a na kome radi migrirana baza.

Praksa pokazuje da je najbolje prikupiti što više informacija različite vrste u fazi planiranja. Ove informacije je potrebno detaljno analizirati kako bi se predvideli svi tehnički problemi koji mogu nastati tokom procesa migracije. Sva zapažanja je potrebno dokumentovati a kasnije vršiti akcije na osnovu zapisanih beleški kako bi se smanjio rizik.

2.3 Analiza i dizajn

Faza analize i dizajna se načešće sastoji od utvrđivanja detalja implementacije na ciljni sistem za upravljanje bazom podataka. Zbog razlika koje se javljaju u implementiranim tipovima podataka, bezbedonosnim ulogama i privilegijama, upravljanjem transakcijama i SQL kodu veoma je važno razviti plan migracije koji uzima u obzir odgovarajuće karakteristike i funkcionalnosti na ciljnom sistemu za upravljanje bazom podataka. Mora se takođe uzeti u obzir koliko nove karakteristike i funkcionalnosti koje poseduje ciljna platforma utiču na promenu koda aplikacija na izvornom sistemu ili na promenu kvaliteta podataka. Sada će ukratko biti opisani najvažniji problemi koji moraju biti prepoznati prilikom migracije na ciljni sistem.

Koncept organizacije šema u bazi za različite RSUBP–ove – Veoma je važno razmotriti na koji način će se preslikati šema izvorne baze i da li će izabrani način preslikavanja imati uticaj na aplikacije i SQL naredbe koje su njen sastavni deo. RSUBP-ovi se međusobno razlikuju u zavisnosti od toga kako su šeme, korisnici, i objekti organizovani u njihovim bazama podataka. Mnogi sistemi za upravljanje bazom

podataka, kao što su DB2, MS SQL Server i Sybase, podržavaju više baza kojima odgovara jedan proces za upravljanje bazom podataka(instanca). U svakoj od tih baza objekti mogu biti organizovani u smislu šema. S druge strane RSUBP kao što je Orakl podržava samo jednu bazu podataka po instanci. Ove razlike koje se javljaju na nivou šema između Orakl i DB2 sistema za upravlajnje bazom podataka mogu da uzrokuju koliziju objekata u ciljnoj Orakl bazi jer DB2 sistem dozvoljava da postoje objekti različite strukture sa istim imenom unutar različitih baza kojima upravlja jedna instanca.

Konvencije za davanje imena objektima baze – Jedan od najvećih problema koji se javljaju pri migraciji baze odnose se na konvenciju davanja imena objektima. Ovde treba posebnu pažnju obratiti na sledeće:

- **Korišćenje rezervisanih reči** - RSUBP-ovi se značajno međusobno razlikuju u smislu šta koji od njih podrazumeva pod rezervisanom reči (reči koje ne mogu biti korišćene kao imena objekata u bazi ili kolona u tabelama baze). Ovo se javlja kao posledica toga da svaki sistem za upravljanje bazom koristi svoj interni spisak rezervisanih reči. Za vreme migracije moguće je da pojedine tabele izvorne baze sadrže kolone koje su u skladu sa restrikcijama na ciljnom sistemu. Ovakve situacije se prevazilaze tako što alati za migraciju najčešće imaju u sebi ugrađen mehanizam uz pomoć koga konvertuju imena objekata koristeći predefinisanu konvenciju davanja imena koja je prihvatljiva za ciljni sistem.

- **Dužina imena objekata** – RSUBP Orakl dozvoljava maksimalnu dužinu imena objekta u šemi od 30 karaktera. S druge strane RSUBP DB2 dozvoljava veličinu i do 128 karaktera. Tako da ukoliko ime nekog objekta nije u skladu sa predefinisanom dužinom na ciljnom sistemu onda taj objekat mora biti preimenovan u skladu sa ograničenjem. Alati za migraciju mogu da identifikuju ovakve slučajeve i naprave izveštaj u kome su imena objekata čija dužina nije u skladu sa ograničenjem.

- **Korišćenje specijalnih karaktera u nazivima objekata baze** – Takođe, svaki sistem za upravljanje bazom podataka ima svoju internu politiku u smislu koji specijalni karakteri mogu da se koriste u nazivima objekata baze. Tako na primer korišćenje specijalnog karaktera # kao prvog u nazivu nekog objekta Orakl baze nije dozvoljeno što nije slučaj za druge baze. Najčešće se ovakvi objekti preimenuju za vreme konverzije što može prouzrokovati potrebu za promenom koda aplikacija koje koriste ove objekte.

Preslikavanje tipova podataka – U svakoj bazi postoji mnoštvo tipova podataka koji obuhvataju numeričke, karakter, velike objekte, XML i vremenske podatke. Prethodno navedeni tipovi podataka u različitim bazama su najčešće standardni ali i pored toga se međusobno značajno razlikuju u smislu ograničenja na dužinu podatka i dozvoljenu preciznost. Najčešći problemi se javljaju kada želimo da konvertujemo tipove sa izvornog sistema koji ne postoje na ciljnom sistemu. Ovaj problem se prevazilazi tako što se nedostajući tipovi konvertuju u neke druge postojeće i pritom odgovarajuće tipove na ciljnom sistemu.

Mehanizmi zaključavanja – Pristup prilikom zaključavanja tokom konkurentnog čitanja podataka u Orakl i DB2 bazi je različit. Na ove razlike posebno treba obratiti pažnju prilikom migracije aplikacija koje rade nad bazom. Ukoliko postoje razlike u mehanizmima zaključavanja na novom sistemu, aplikacija neće ispravno raditi i njena funkcionalnost će biti narušena.

Korišćenje praznih niski – Mnogi RSUBP-ovi, među kojima i DB2, podržavaju prazne niske čija je dužina nula ali se ne posmatraju kao NULL vrednosti već kao prazni stringovi. U Orakl bazi ne postoji podrška za

rad sa praznim stringovima. Potrebno je predvideti vrednosti praznih stringova i prepoznati ih kao različite od NULL vrednosti.

Sigurno je da postoje i drugi problemi na nivou dizajna baze koji se mogu pojaviti tokom migracije sa izvornog sistema na ciljni sistem. Određene specifičnosti izvorne baze, u smislu na koji način je određena karakteristika ili funkcionalnost njoj svojstvena iskorišćena da olakša određeni poslovni proces, moraju biti prepoznate pre migracije. Pažljivo razmatranje i rešavanje ovih problema je od velike važnosti za prevazilaženje prepreka na putu tokom migracije kao i u fazi posle migracije.

2.4 Faza migracije

Vreme potrebno da bi se izvršila migracija šeme, podataka, objekata baze, aplikacija iz jedne baze u drugu predstavlja značajan deo količine vremena koju je potrebno utrošiti na celokupni projekat migracije od faze planiranja do produkcije. Najveći proizvođači RSUBP-ova najčešće obezbeđuju softverske alate koji olakšavaju i automatizuju proces robusnih migracija. Bez obzira na nivo faktora uspešnosti prilikom automatizacije poslova migracije vrlo često su potrebne ručne intervencije. Poslovi migracije baze mogu biti podeljeni u sledeće kategorije:

- Migracija šeme
- Migracija podataka
- Migracija procedura, funkcija, okidača
- Migracija aplikacija baze
- Migracija skriptova za administraciju

Pored svih prethodno navedenih poslova migracije, posao migracije aplikacija zahteva najviše ručnih intervencija. U daljem tekstu će biti opisana najbolja praksa prilikom izvršavanja prethodno navedenih poslova.

2.4.1 Migracija šeme baze

Migracija šeme baze podataka u suštini obuhvata migraciju tabela, indeksa i pogleda u bazi. Relacione baze su međusobno slične u smislu kako su njihovi podaci organizovani u tabelama i indeksima, ali su različite u smislu određenih koncepata definisanih nad tabelama i indeksima koji su dizajnirani kako bi se poboljšale performanse i olakšao razvoj. Mnogi alati za migraciju mogu konvertovati šemu baze relativno brzo i precizno. Takođe šema na ciljnoj bazi može biti napravljena korišćenjem alata za modelovanje kao što je Erwin. U daljem tekstu će biti izloženo šta je to najvažnije što treba razmotriti prilikom migracije šeme baze.

Potpunost migrirane šeme – Neophodno je obezbediti da svi objekti šeme sa izvorne baze migriraju na ciljnu bazu. Veoma je često potrebno migrirati više šema da bi podržale rad neke aplikacije. Postojanje zavisnosti između više šema može rezultovati greškama prilikom pravljenja šeme na ciljnoj bazi, jer pojedine zavisnosti mogu nedostajati kada određena šema migrira. Kada su šeme na ciljnoj bazi napravljene za sve objekte koji su konvertovani prilikom migracije potrebno je ponovo proveriti da li su uspešno migrirani.

Pravljenje korisničkih naloga i pridruživanje odgovarajućih uloga korisnicima – Odgovarajuće uloge i privilegije nad objektima moraju biti ispravno dodeljene korisnicima. Privilegije nad šemom i objektima u njoj mogu biti grupisane u uloge i dodeljene korisnicima ako je to potrebno. Dodeljivanje uloga i njihovo dodeljivanje korisnicima olakšava proces upravljanja privilegijama nad objektima šeme.

Izmena naziva objekata - Promena imena bilo kog objekta u bazi mora biti identifikovana i ukazana svim ostalim članovima tima za migraciju tako da oni mogu napraviti odgovarajuće izmene nad aplikacijama i ostalim komponentama baze.

Particionisane tabele – Velike tabele mogu biti particionisane u manje segmente kako bi se postigle bolje performanse na taj način što optimizator upita zanemaruje određene particije koje nisu pogođene upitom. Ovakav pristup smanjuje količinu podataka koja je zahvaćena obradom. Da li će određena tabela biti particionisana ili ne zavisi od količine podataka u njoj, zahtevanih performansi, zahteva koji ukazuju na koji način će se upravljati sa podacima u toj tabeli. Mnoge RSUBP-ovi daju podršku za particionisanje tabela. Razlike među njima postoje u pristupu, to jest na koji način omogućavaju da se vrši particionisanje tabela to jest da li se particionisanje vrši po range-u, listi vrednosti, hash-u ili je u pitanju kompozitni pristup. Alati za migraciju generalno migriraju samo tabele ali ne i njihove definicije koje ukazuju na koji način su te tabele particionisane. Dakle, potrebno je posle migracije strukture tabela a pre migracije podataka, definisati jasnu strategiju na osnovu koje će se izvršiti particionisanje migriranih tabela.

Korišćenje klasterovanih indeksa – Koncept klasterovanih indeksa nije isti u Orakl i DB2 bazi. Klasterovani indeks definisan nad tabelom u DB2 bazi ukazuje na koji način će podaci biti organizovani u njoj. U Orakl bazi pod klasterovanim indeksom se smatra svaki indeks definisan nad particionisanom tabelom. Da bi se prevazišle ovakve razlike potrebno je iskoristiti slične funkcionalnosti koje nudi novi sistem na kome radi migrirana baza. U nekim situacijama je dovoljno da bude približno pokrivena funkcionalnost koja je postojala na izvornom sistemu.

2.4.2 Migracija podataka

Posle migracije šeme, za slučaj da se radi o velikoj količini podataka koje treba migrirati, najpre se odredi reprezentativan uzorak podataka u izvornoj bazi koji treba da migrira na ciljnu bazu kako bi se omogućilo testiranje i utvrdilo da li su skriptovi za migraciju podataka ili odabrani alati ispravno konfigurisani. Najčešći pristup pri migraciji podataka je korišćenje skriptova koji koriste alate za eksport podataka iz izvorne baze i učitavanje podataka u ciljnu bazu.

Bez obzira da li se koriste skriptovi ili alati za migraciju, migracija velikih baza zahteva precizno i sveobuhvatno planiranje poslova koje treba izvršiti. Kada se migriraju velike količine podataka, na primer nekoliko terabajta podataka, veoma je važno imati ispravnu strategiju migracije, odgovarajuće alate, i što je najvažnije prepoznati prave tehnike koje će biti korišćene kao što su particionisanje i kompresija. Proces migracije velikih baza je ispunjen izazovima u smislu vremenskih rokova i raspoloživih sistemskih resursa. Sledeće strategije mogu optimizovati procese ekstrakcije, transformacije i punjenja podataka:

- Paralelna ekstrakcija podataka iz izvorne baze
- Paralelno punjenje podataka u ciljnu bazu
- Korišćenje procesa iz više niti za učitavanje podataka

- Izbegavati održavanje indeksa tokom procesa punjenja podataka

Za migraciju podataka je potrebno manje vremena nego za migraciju i testiranje aplikacija. Poslovi migracije podataka u zavisnosti na koji način se vrše, mogu biti svrstani u tri kategorije:

Offline **migracija podataka** – U ovom slučaju podaci najpre se izdvajaju u nestruktuirane datoteke a potom se ciljna baza napuni tim podacima koristeći alate ili skriptove za migraciju. Zbog toga što su proces izdvajanja podataka i proces učitavanja podataka međusobno odvojeni i nezavisni poslovi, učitavanje podataka se može odraditi i posle određenog vremenskog roka nezavisno od trenutka izdvajanja podataka. Ovo se obično postiže korišćenjem skriptova ili alata koje obezbeđuju proizvođači RSUBP-a kao što su Orakl SQL Loader odnosno LOAD/UNLOAD pomoćni alati u sklopu IBM DB2 sistema za upravljanje bazom podataka.

Online **migracija podataka** – Ukoliko se migracija podataka vrši na ovaj način onda to uključuje aktivnu vezu ka izvornoj i ciljnoj bazi koristeći JDBC (Java Database Conectivity) ili ODBC (Open Database Conectivity) drajvere. Kao i što samo ime sugeriše, tokom migracije podataka, obe baze su dostupne za uspostavljanje veze. Pri ovom načinu migracije dostupnost mreže preko koje ove dve baze komuniciraju je obavezna. Ovaj mod se najčešće koristi za male baze kada posao učitavanja podataka nije zahtevan. Pošto ovaj pristup zahteva dosta resursa u smislu potrebne memorije i performansi procesora, on nije preporučen za slučaj da se radi migracija velike količine podataka.

Praćenje promena nad podacima (Changed data capture CDC) – CDC uključuje praćenje promena nad podacima u izvornoj bazi a potom periodično repliciranje tih promena nad podacima u ciljnoj bazi. Ovo je veoma koristan metod za migraciju velikih baza podataka kada nije dozvoljen veći vremenski zastoj u procesu migracije podataka. CDC može biti implementiran na dva načina:

- **Pretraživanje logova** – Ovo je najčešće korišćena tehnika za implementaciju CDC-a. Pretraživanje log datoteka uključuje čitanje *online* ili arhiviranih transakcionih logova iz baze a potom izdvajanje transakcija iz njih onim redosledom kako su izvršavane na izvornoj bazi. Kada su sve promene nad podacima u izvornoj bazi sakupljene, najpre se smeštaju u privremenu datoteku odakle mogu biti odmah ili posle izvesnog vremena izvršene nad ciljnom bazom. Ovaj metod je veoma popularan jer ne narušava performanse baze i aplikacija i veoma je lak za pripremu i izvođenje.

- **Korišćenje okidača** – Ovaj metod podrazumeva da su okidači implementirani, pri čemu je njihova uloga da prate promene nad podacima i da ih upisuju u privremene tabele odakle se promene dalje prenose na ciljnu bazu. Definisanje novih okidača nad bazom je veoma zahtevan posao koji je ispunjen problemima različite vrste. Pored toga ovaj metod ima negativan uticaj na performanse baze i ne spada u popularne metode za praćenje promena nad podacima.

2.4.3 Migracija procedura, funkcija i okidača

Jedan od zahtevnijih poslova prilikom migracije baze jeste konverzija procedura, funkcija i okidača koji se u relacionim bazama koriste za implementaciju poslovne logike. Česta je situacija da programeri koriste intezivno procedure, funkcije i okidače u kojima su implementirane kako jednostavne tako i složene operacije. Najvažniji zadaci koji se vezuju za migraciju ovih objekata baze su:

Čišćenje i optimizovanje koda - Alati za migraciju daju relativno dobru podršku za migraciju procedura, funkcija i okidača. Posle migracije preporučeno je testiranje ovih objekata zbog provere preciznosti, efikasnosti i mogućih grešaka u konvertovanom kodu. Pošto je moguće implementirati jednostavne poslovne zahteve na mnogo načina, ovaj pristup otežava alatu za migraciju da optimizuje sve moguće korišćene tehnike kodiranja u konvertovanom kodu. Procedure i funkcije sa dosta linija koda moraju posle migracije biti proverene i testirane u smislu korišćenja pojedinih objekata i funkcionalnosti baze kao i zbog sprovođenja prakse optimizacije konvertovanog koda.

Rukovanje greškama – Automatski izvođena migracija često nije u stanju da obuhvati obradu koda koji se odnosi na rukovanje greškama u ugnježdenim procedurama. Samim tim tokom migracije aplikacija je veoma važno obratiti posebnu pažnju na rukovanje greškama naročito pri pozivu ugnježdenih procedura.

Intezivno korišćenje privremenih tabela – Programeri često koriste privremene tabele kako bi pojednostavili upite odnosno izbegli pisanje složenih upita koji uključuju više tabela. Ranije verzije RSUBP-ova su imale čak i restrikcije na broj tabela koje se mogu efikasno spojiti u upitima. Alati za migraciju vrše direktno preslikavanje privremenih tabela prilikom njihove migracije sa jedne baze na drugu. Prilikom migracije procedura koji koriste veliki broj privremenih tabela česta je situacija ispitivanja da li se pojedine privremene tabele mogu zanemariti kako bi se dobio što jednostavniji i razumljiviji kod. Ovo se posebno odnosi na najčešće korišćene procedure koje imaju velike zahteve u smislu performansi gde je potrebno detaljno ispitati kako bi se eliminisale one privremene tabele koje više nisu potrebne u novom okruženju.

Konvertovanje procedura u funkcije – Neki RSUBP–ovi kao što je Orakl ne dozvoljavaju korišćenje RETURN klauze u kodu procedure kako bi se vratila povratna vrednost pozivaocu procedure. Ova klauza je jedino dozvoljena da se koristi u kodu funkcija ali ne i u kodu procedura. Da bi se prevazišle razlike u sintaksi procedura i funkcija koje dozvoljavaju različiti RSUBP-ovi umesto RETRURN klauze se koriste OUT parametri kako bi se vratila povratna vrednost pozivaocu funkcije. Dešavaju se situacije da prilikom migracije jedan od zahteva bude i konverzija funkcija u procedure sa jednim OUT parametrom.

2.4.5 Migracija skriptova za poslove administracije

Administratori baze podataka razvijaju svoje skriptove kako bi obavljali poslove administracije i postigli izvestan stepen automatizacije prilikom obavljanja ovih poslova. Korišćenje skriptova radi obavljanja poslova administracije baze mogu da zakomplikuju stvari onda kada administrator koji ih je razvio napusti radnu organizaciju, jer novi administratori nemaju potrebno znanje da bi te skriptove efektivno koristili. Skriptovi koji se koriste za praćenje performansi, održavanje spiska korisnika, održavanje objekata baze se najčešće ne migriraju, jer najčešći razlozi migracije na neku drugu platformu jeste i to što ona pruža udobniji rad administratorima kroz svoje alate za administraciju koji pružaju jednostavan korisnički interfejs za obavljanje ovih poslova. Takođe isto važi i za skriptove koji vrše izdvajanje ili učitavanje podataka. Za ovakve poslove RSUBP-ovi takođe imaju alate koji imaju iste funkcionalnosti kao i skriptovi.

2.4.6 Migracija aplikacija

Glavni izazov koji se javlja prilikom migracije aplikacija jedne baze na drugu platformu jeste identifikovanje segmenata koda koje je potrebno izmeniti kako bi aplikacija ispravno radila u novom okruženju. U klijent/server okruženju gde je logika aplikacija pokrivena hiljadama linija koda, veliki je

izazov identifikovanje delova koda i SQL naredbi ugnježdenih u njima koje je potrebno izmeniti. Posao migracije aplikacija jedne baze se može izvršiti na više načina, od ručnog prepisivanja koda aplikacija i prepoznavanja nekompatibilnosti koje se mogu pojaviti u novom okruženju do upotrebe alata za konverziju aplikacija. Jedan od pristupa se ogleda i u razvoju skriptova u jezicima kao što je Perl koji automatski identifikuju segmente koda koje je potrebno izmeniti. Svaka od prethodno navedenih opcija ima svoje prednosti i mane. Najbolji pristup je dobro poznavanje svih opcija za migraciju i na osnovu toga odabrati onu koja najviše izlazi u susret poslovnim i tehničkim zahtevima.

Izmene koje je potrebno izvršiti na kodu kako bi aplikacija migrirala na novu platformu mogu biti kategorizovane na osnovu uloge koda koji se menja u okviru aplikacije ili na osnovu njegove zavisnosti od izabrane platforme na koju aplikacija migrira. U zavisnosti na šta se odnose ovde razlikujemo nekoliko tipova izmena:

Konekcija prema bazi – Svi delovi koda aplikacije koji se odnose na uspostavljanje konekcije ka novoj bazi moraju biti izmenjeni, jer svakoj bazi odgovara poseban skup parametara koje je potrebno u vidu stringa za konekciju proslediti drajverima koji omogućavaju uspostavljanje konekcije.

Ugnježdene SQL naredbe – U kodu aplikacija se može nalaziti veliki broj ugnježdenih SQL naredbi, uključujući kako statičke tako i dinamičke SQL naredbe. Takođe, kod može da sadrži i veliki broj poziva procedura ili funkcija. Ovde je potrebno posebnu pažnju obratiti na nadogradnje SQL jezika koje se razlikuju između različitih proizvođača RSUBP-ova i koje mogu uzrokovati nekompatibilnost migriranih aplikacija u novom okruženju.

Segment koda aplikacije – Pored izmena koje se odnose na ugnježdene SQL naredbe, postoje i one izmene koje se odnose na segmente koda aplikacije prouzrokovane su tipom drajvera koji se koristi za pristup bazi. U ove segmente koda spadaju oni koji upravljaju vraćenim skupom rezultata iz baze posle izvršene neke SQL naredbe i oni segmenti iz kojih se vrši poziv neke procedure ili funkcije.

Implementacja API-ja – Različiti RSUBP-ovi se međusobno razlikuju u odnosu na način na koji implementiraju API koji olakšava pristup tokom obrade podataka iz baze i kontrole transakcija iz aplikacije. Kao posledica migracije baze na novu platformu javlja se potreba i za promenom ovih API-ja.

Rukovanje greškama – Svi RSUBP-ovi se međusobno razlikuju po načinu na koji definišu određene poruke koje se javljaju kao posledica određenih grešaka u radu aplikacije. Kako ne bi došlo do zabune ove poruke se moraju prepraviti u kodu aplikacije u smislu da budu identične onim porukama koje bi vratio novi RSUBP na koji migrira baza.

Optimizacija performansi – Kao posledica promene platforme na kojoj radi baza javlja se očekivanje poboljšanja performansi. Da bi se postigao takav rezultat potrebno je izvršiti određena podešavanja drajvera i optimizaciju delova koda aplikacije.

Svi prethodno navedeni tipovi izmena imaju zajedničku karakteristiku a to je da zahtevaju veliku količinu vremena i truda.

2.5 Testiranje

Značajan deo vremena i napora je potrebno uložiti u testiranje aplikacija i baze posle izvršenog procesa migracije. Faza testiranja u jednom projektu migracije obično podrazumeva poslove kao što su

ispitivanje ispravnosti podataka, testiranje poslovne logike u migriranim procedurama, funkcijama i okidačima, testiranje interakcije migriranih aplikacija sa novom platformom i testiranje migriranih skriptova za održavanje baze. Neki od ovih poslova testiranja mogu biti relativno lako izvršeni uz pomoć alata za testiranje ili uz pomoć skriptova. U daljem tekstu će biti dat pregled različitih pristupa za svaki od poslova u fazi testiranja.

Provera ispravnosti podataka – Najlakši način da se proveri ispravnost migriranih podataka jeste da se detaljno prati proces migracije podataka kako bi se utvrdilo da ne postoje greške tokom procesa. Čak i kada alati za migraciju podataka ne prijavljuju greške tokom procesa moguće je da se javljaju problemi kao što je automatsko podkresivanje decimalnih i karakter podataka koje nastaje kao posledica nedozvoljene veličine podatka u odnosu na dozvoljenu veličinu tipa kolone na ciljnom sistemu.

Testiranje procedura, funkcija i okidača - Najčešće su ovi objekti pojedinačno testirani u vreme njihove migracije radi provere njihove sintakse i semantike. U svakom slučaju, posle migracije čitave baze neophodno je tesirati međuzavisnosti njenih objekata na ciljnom sistemu. Testiranje funkcija i procedura se vrši za različite ulazne parametre. Tokom testiranja veoma je teško pronaći sve moguće vrednosti za ulazne parametre koje mogu prihvatiti procedure i funkcije. Postojanje unapred pripremljenih svih mogućih slučajeva koji su dokumentovani u vidu skriptova u značajnoj meri olakšava ovaj posao testiranja.

Testiranje aplikacija – U mnogim organizacijama testiranje aplikacija razvijenih od strane njihovih programera vrše testeri uz pomoć specijalno namenjenih korisničkih interfejsa za testiranje a na osnovu predefinisanih i dokumentovanih slučajeva za testiranje. Proizvođači RSUBP-a najčešće nude alate za testiranje aplikacija koji mogu biti korišćeni za testiranje funkcionalnosti aplikacija, za testiranje njihove skalabilnosti i performansi pri čemu ti alati pružaju pomoć pri definisanju novih slučajeva. Pojedini alati mogu da snime sve moguće načine interakcije aplikacije sa bazom pa je na osnovu prikupljenih informacija moguće definisati dobre slučajeve za testiranje.

Testiranje skriptova za održavanje baze – Veoma je važno izvršiti testiranje skriptova koji vrše poslove bekap-a i oporavka baze. Mnogi poslovi bekap-a i oporavka baze mogu biti automatizovani a skriptovi koji vrše te poslove mogu biti ponovno korišćeni u nekim drugim situacijama. Testiranje ovih skriptova se vrši ručno i potrebno je da se ovaj posao izvrši u izolovanom okruženju. Novije verzije različitih platformi nude alternativne načine preko kojih se mogu obavljati inkrementalne operacije bekap-a i oporavka baze.

2.6 Faza implementacije

Mnogi poslovi koje bi trebalo izvršiti u ovoj fazi su zapravo definisani u fazi planiranja i fazi analize i dizajna. Tokom ove dve faze je izvršena procena arhitekture ciljnog sistema i svih neophodnih softverskih komponenti koje će se koristiti u novom sistemu. Poslovi iz faze implementacije se zapravo obavljaju na samom početku projekta migracije, jer mnoge organizacije žele da prate poslovnu praksu u pogledu snabdevanja novim softverom i hardverom.

Pošto se u okviru ove faze vrši nabavka novog softvera i hardvera, dodatne procene moraju biti definisane kako bi se RSUBP ispravno konfigurisao. Ove procene se odnose na konfigurisanje deljenog prostora za čuvanje podataka, zatim mreže u kojoj će se komunicirati sa bazom itd. Najčešći poslovi koji se izvršavaju u toku ove faze su:

Konfiguracija hardvera – Ovaj posao uključuje konfigurisanje servera na kome se nalazi baza, prostora za podatke na disku, konfigurisanje mrežnog okruženja u kome radi baza. Pažnja mora biti usmerena na procenu potrebnog hardvera i memorijskog prostora u skladu sa radnim opterećenjem baze.

Instalacija i konfiguracija softvera – Ovaj posao se primarno sastoji od instalacije i konfiguracija RSUBP-a na kome će se nalaziti ciljna baza, kao i instalacije šeme na ciljnoj bazi i njeno puštanje u produkciju i povezivanje sa aplikacijama koje koriste njene objekte. Posle podešavanja šeme, sigurnosne uloge i privilegije moraju biti dodeljene korisnicima aplikacija.

Inicijalno učitavanje podataka – Posle definisanja šeme u produkcionom okruženju, sledeći posao je učitavanje najreprezentativnijih podataka sa izvornog sistema. Podaci se učitavaju uz pomoć skriptova ili alata za učitavanje podataka. Veoma je važno posle inicijalnog učitavanja podataka a pre dozvole korisnicima da rade sa bazom, utvrditi i da li su svi potrebni indeksi napravljeni.

Praćenje promena na izvornom sistemu i njihovo prepisivanje na ciljni sistem – Za one baze u čijem radu ne sme da bude zastoja i za koje važi da stalno moraju biti dostupne, veoma je važno da sve promene koje nastaju u toku vremena učitavanja podataka i faze prelaska budu zabeležene i kasnije izvršene na ciljnoj bazi. Ovaj pristup je veoma bitan kako bi se izbegao nedostatak neke promene nad podacima u bazi, od strane izvršene transakcije koja bi mogla uticati na njihov integritet u novom okruženju. Takođe vendori i za ovaj posao nude bogat skup alata. Njihov rad se sastoji u praćenju promena na izvornoj bazi i njihovo repliciranje na ciljnoj bazi. Ovaj proces se obavlja u nekoliko koraka:

1. Podešavanje procesa praćenja promene podataka na izvornom sistemu
2. Izdvajanje podataka sa izvornog sistema
3. Učitavanje prethodno izdvojenih podataka u bazu na ciljnom sistemu
4. Primena svih promena na ciljnom sistemu
5. Puštanje baze u produkciju

2.7 Post – produkciona podrška

Veoma je česta praksa da poslovne organizacije kod kojih je implementiran novi sistem za upravljanje bazom podataka unajmljuju podršku, po puštanju sistema u produkciju, od strane onih koji su bili uključeni u proces migracije baze sa starog na novi sistem. Njihova uloga je da predvide i pronađu sve moguće loše scenarije koji bi se mogli pojaviti od trenutka kada je sistem krenuo u produkciju. Mogući problemi koji se mogu pojaviti su:

- Problem slabog poznavanja novog okruženja od strane IT osoblja.
- Problemi vezani za performanse pri izvršavanju upita.
- Aplikacije i programi se ne izvršavaju kako je očekivano zbog nedostatka neke njihove funkcionalnosti posle migracije ili zbog čudnog ponašanja u različitim situacijama. Ovo se

najčešće dešava kada neka funkcionalnost aplikacije nije migrirala a kasnije aplikacija nije testirana.

- Problemi vezani za reprezentaciju podataka u migriranim aplikacijama.
- Vreme potrebno za administratore i programere kako bi postali potpuno familijarni sa novim okruženjem.

2.8 Rezime

Primenom metodologije, kod velikih informacionih sistema migracija baze može da se svede na niz potpuno definisanih i kontrolisanih aktivnosti koje garantuju uspešno izvršavanje i najsloženijih migracija. Fazni pristup opisan u metodologiji je primenljiv i na slučajeve migracije koji su opisani u ovom radu.

3. Migracija DB2 baze na Orakl RSUBP

U osnovi svakog procesa migracije, bez obzira na koji ciljni RSUBP se baza migrira, želja je da se migriraju objekti šeme i podaci u što manjem vremenskom intervalu pri čemu je cena troškova pri obavljanju takvog posla svedena na minimum. Radi postizanja ovih ciljeva svi proizvođači sistema za upravljanje bazom podataka, pa samim tim i Orakl, obezbeđuju skup softverskih alata čijom upotrebom se prevazilaze vremenske, finansijske kao i barijere rizika koje se mogu javiti tokom izvršavanja procesa migracije.

U zavisnosti od konkretnog projekta migracije odabere se skup alata kojim će se izvršavati određeni poslovi migracije. Mnogo faktora može uticati na složenost projekta migracije. Neki od najvažnijih su:

- Veliki broj objekata baze, odnosno šeme.
- Složenost poslovne logike sačuvane u bazi kroz procedure, funkcije, okidače i aplikacije koje rade nad bazom.
- Složenost procedura, funkcija, okidača, pogleda u smislu broja linija koda kojim su implementirane.
- Količina podataka u bazi, koja u slučajevima da se radi o skladištima podataka može biti i nekoliko terabajta.
- Kritično vreme dostupnosti baze prilikom migracije.
- Složenost dokumentovanih slučajeva za testiranje migriranih aplikacija.

Svi prethodni faktori mogu uticati na specifičan odabir alata za migraciju koji određuje na koji način će se sprovoditi projekat migracije. Ovde razlikujemo dva pristupa:

- Brzi pristup miraciji koji podrazumeva da alati direktno komuniciraju sa izvornom bazom, čitaju definicije tabela i migriraju podatke kroz mrežu koja povezuje dve baze. Ovaj pristup podrazumeva korišćenje API-ja za pristup bazi. Svi poslovi migracije se odigravaju automatizovano jedan za drugim bez mogućnosti vraćanja i vršenja ispravki prethodno urađenog koraka.
- Standardni pristup migraciji koji podrazumeva izvršavanje poslova migracije nezavisno jedan od drugog. Da bi se jedan posao migracije mogao udraditi potrebno je samo da su njegovi svi uslovni poslovi izvršeni pri čemu on ne mora biti izvršen neposredno posle njih. Tako na primer, najpre se izvrši posao prebacivanja podataka iz izvorne baze pomoću alata za migraciju podataka na deljeni disk a potom se kasnije vrši odvojeno učitavanje podataka u Orakl bazu.

Proces migracije je dosledan bez obzira sa kog RSUBP-a se migrira baza na Orakl RSUBP. Tipičan proces migracije na Orakl RSUBP uključuje sledeće korake:

- Prikupljanje metapodataka svih objekata šeme izvorne baze.
- Konverzija objekata izvorne baze na Orakl RSUBP koristeći prethodno obrađeni model metapodataka.
- Pravljenje šeme u ciljnoj Orakl bazi.

- Prebacivanje izvornog koda svih klijentskih aplikacija u novo okruženje sa Orakl bazom kako bismo bili sigurni da sve SQL naredbe i API-ji za komunikaciju sa bazom rade ispravno.
- Migracija podataka iz izvorne baze u ciljnu Orakl bazu.
- Testiranje migriranih objekata baze, podataka i aplikacija.
- Optimizacija Orakl baze.
- Puštanje migrirane baze u produkciju.

Veoma je bitna uspešnost pojedinih zadataka koji se obavljaju tokom procesa migracije. U takve poslove spadaju:

- Migracija poslovne logike implementirane u izvornoj bazi koja migrira na Orakl RSUBP.
- Migracija podataka.
- Migracija klijentskih aplikacija koje treba da rade u novom okruženju.
- Testiranje migriranih aplikacija i poslovne logike koja je implementirana u njima.

Nezavisno od toga koji je nivo automatizacije obezbeđen odabirom određenog skupa alata za migraciju u svakom projektu migracije od posebnog značaja je sistematski pristup prilikom testiranja migrirane baze na Orakl ciljnu platformu.

3.1 Migracija DB2 baze pomoću Orakl SQL Developer alata

Migracija neke relacione baze na Orakl RSUBP uglavnom uključuje korišćenje migracionog alata koji migrira ne samo objekte šeme već takođe i poslovnu logiku implementiranu u objektima baze kao što su procedure, funkcije, okidači i pogledi. Vremenom su svi proizvođači RSUBP-ova pa samim tim i Orakl nastojali da obezbede migracione alate kojima su nastojali da pokriju sve moguće poslove migracije i da obezbede što bolji kvalitet konverzije.

Kompanija Orakl nudi integrisani alat SQL Developer za migraciju baze na Orakl platformu sa drugih platformi kao što su Microsoft SQL Server, Sybase, MySQL, Microsoft Access, IBM DB2 LUW i Teradata.

SQL Developer predstavlja grafički alat koji omogućava ugodan rad administratora i programera nad bazom tako što povećava produktivnost i uprošćava njihove zadatke rada sa bazom. Koristeći SQL Developer korisnici mogu da pretražuju, definišu, vrše izmene nad objektima baze, pokreću SQL upite, menjaju i interaktivno prolaze kroz PL/SQL kod i imaju pristup obimnom spisku predefinisanih izveštaja.

Zahvaljujući SQL Developer-u, moguće je uspostaviti konekciju i ka ne-Orakl bazama kao što su Sybase, MySQL, Microsoft SQL Server i IBM DB2. Kada je konekcija uspostavljena, alat obezbeđuje poseban interfejs za migriranje prethodno navedenih baza na Orakl RSUBP. U zavisnosti o kojoj se izvornoj bazi radi, SQL Developer automatski može da konvertuje tabele, okidače, procedure, funkcije i druge relevantne objekte na Orakl bazu. Kada je po završetku automatizovanog procesa migracije baza na Orakl RSUBP-u napravljena, SQL Developer može da pruži podršku i pri migraciji podataka iz izvorne ka ciljnoj bazi.

3.1.1 Uspostavljanje konekcije pomoću SQL Developer-a

Interfejs za migraciju predstavlja integralni deo SQL Developer alata, i dostupan je odmah po instalaciji. Pre nego što se započne proces migracije potrebno je uraditi sledeće poslove:

- Uspostaviti konekciju ka Orakl bazi.
- Uspostaviti konekciju ka ne-Orakl bazi. Ovde je potrebno napomenuti da je ugrađena mogućnost uspostavljanja konekcije ka Orakl i Microsoft Access bazi u Orakl SQL Developer-u. Uspostavljanje konekcija ka Microsoft SQL Server, Sybase, IBM DB2 LUW, Teradata i MySQL bazama zahteva odgovarajući JDBC drajver.
- Definisanje repozitorijuma za migraciju. SQL Developer, da bi izvršio poslove migracije, zahteva postojanje repozitorijuma u koji će biti smeštene sve informacije prikupljene tokom procesa migracije. Kada je migracioni repozitorijum definisan u okviru konekcije, SQL Developer automatski definiše tabele i pakete potrebne za prikupljanje informacija.

3.1.2 Proces migracije pomoću SQL Developer-a

Projekat migracije kao i svi drugi IT projekti zahteva analizu, planiranje i testiranje. Potrebno je napomenuti da uz pomoć SQL Developer-a nije moguće izvršavanje poslova iz prethodno navedenih faza u projektu migracije. Zapavo, glavni fokus pri korišćenju ovoga alata u procesu migracije jeste na što većoj automatizaciji poslova u samom procesu migracije.

SQL Developer pruža podršku za dva pristupa migracije sa ne-Orakl baza na Orakl platformu:

- Standardnu migraciju
- Brzu migraciju

3.1.3 Standardna migracija

Standardni pristup migracije, koji se sastoji iz četiri zasebne faze, je više preporučen od brzog pristupa migracije, jer dozvoljava intervenciju posle svake faze gde se potencijalni problemi mogu otkloniti. Ovaj način izvršavanja migracije omogućava izvršavanje svakog koraka pojedinačno, tako da se preostali koraci mogu naknadno izvršiti. Takođe obezbeđuje precizniju kontrolu modifikovanja struktura, brisanja i preimenovanja objekata koji migriraju na Orakl bazu. Kod standardnog pristupa u migraciji SQL Developer deli posao migracije u četiri zasebne faze:

Obrada metapodataka izvorne baze - Prvi korak koji je potrebno izvršiti u projektu migracije jeste procesiranje metapodataka ne-Orakl baze. Posle tog procesa informacije su sačuvane kao model preseka stanja izvorne baze u repozitorijumu za migraciju. Sačuvani model kasnije može biti izmenjen bez bilo kakvog odraza na izvornu ne-Orakl bazu.

Konverzija modela metapodataka - Prethodno napravljeni model se konvertuje u Orakl model. Ovaj model sada predstavlja budući izgled Orakl baze. Novonastali model koji se nalazi u repozitorijumu se takođe može menjati i ne predstavlja kranju bazu koja se dobija procesom migracije.

Pravljenje ciljne baze - Za konvertovani model se potom napravi SQL skript, koji kada se pokrene napravi sve tabele, okidače, procedure, korisnike i druge relevantne objekte izvorne baze.

Prenos podataka - Na kraju, kada imamo model metapodataka izvorne baze, konvertovan Orakl model i napravljenu ciljnu Orakl bazu poslednji korak u procesu migracije jeste prenos podataka između izvorne i ciljne baze, što takođe može biti urađeno uz pomoć SQL Developer-a gde razlikujemo dva pristupa.

- Prvi, takozvani *online* pristup podrazumeva korišćenje uspostavljene konekcije ka ne – Orakl bazi. Ovaj pristup je preporučen kada se vrši migracija malih baza sa malom količinom podataka.
- Drugi, takozvani *offline* pristup podrazumeva korišćenje skriptova napravljenih od strane alata za migraciju. Ovi skriptovi koriste Orakl SQL Loader alat za učitavanje podataka u Orakl bazu.

Tokom procesa migracije SQL Developer omogućuje preko posebnog korisničkog interfejsa pretragu i modifikaciju ne-Orakl baze, njenog modela metapodataka, kasnije konvertovanog modela i na kraju dobijene ciljne baze.

3.1.4 Brza migracija

Opcija brze migracije koju nudi SQL Developer predstavlja zapravo takozvani pristup u jednom koraku. Ovim pristupom korisnici preko interfejsa za migraciju alata definišu parametre koji se tiču ne-Orakl konekcija, konekcije ciljne Orakl baze i informacije vezane za repozitorijum migracije. Zahvaljujući prethodno navedenim informacijama dobijeni su osnovni parametri za podešavanja i moguće je proći kroz korake koji su prethodno opisani u delu za opis standardne migracije a to su: obrada metapodataka izvorne baze, konverzija napravljenog modela metapodataka, pravljenje objekata ciljne baze i na kraju prenos podataka. Ovaj pristup je jedino primenljiv na male baze ili u ulozi testnog zadatka migracije. Ukoliko se pojavi bilo koja greška u toku migracije onda se ceo proces zaustavlja.

3.1.5 *Online* ili *offline* način migracije na Orakl bazu

Moguće je koristiti različite metode prilikom obrade metapodataka, migriranja objekata i podataka iz ne-Orakl baze. Uopšteno govoreći, u zavisnosti od toga da li se radi sa uspostavljenom konekcijom na ne – Orakl bazu ili bez uspostavljene konekcije, migraciju možemo izvršiti preko dve metode:

- *Online* - koristeći direktan pristup ne-Orakl bazi i izvršiti proces migracije preko konekcije napravljene pomoću SQL Developer-a.
- *Offline* - koji se vrši pomoću skriptova koje na ne-Orakl bazi koje napravi alat, a potom izvršavanje tih skriptova nad Orakl bazom.

4. Praktični deo migracije DB2 baze na Orakl RSUBP

U okviru ovog dela biće predstavljen proces migracije DB2 baze na Orakl RSUBP korišćenjem SQL Developer alata za migraciju. Kao primer je korišćena SAMPLE DB2 baza koja dolazi zajedno sa instalacijom DB2 RSUBP-a. Najpre će biti dat prikaz svih komponenti jedne DB2 baze koje je moguće automatski konvertovati i dodatnih pogodnosti koje se dobijaju zahvaljujući ovom pristupu koji se zasniva na korišćenju alata za migraciju. Potom će biti dat prikaz svih pripremnih poslova koje je potrebno udraditi kako bi se alat mogao koristiti u projektu migracije. Na kraju će biti dat prikaz svih koraka kroz proces migracije šeme SAMPLE DB2 baze pomoću SQL Developer-a.

4.1 Migracija DB2 baze na Orakl RSUBP korišćenjem SQL Developer-a

Izvršavanje poslova u okviru projekta migracije se može olakšati faznim pristupom koji obezbeđuje Orakl SQL Developer. Svaka faza u poslu migracije se može izvršiti zasebno ili uz pomoć korisničkog grafičkog interfejsa. Korišćenjem grafičkog migracionog čarobnjaka migracija baze se može izvršiti u interaktivnom modu.

Objekti DB2 baze čije je migriranje na Orakl bazu podržano ovim pristupom jesu:

- Tabele
- Pogledi
- Indeksi
- Korisnici
- Ograničenja

Tehnologije za migraciju SQL Developer alata podržavaju sledeće osnovne mogućnosti prilikom migracije sa IBM DB2 baze:

- Automatsko konvertovanje DB2 tipova podataka koji odgovaraju kolonama tabela u odgovarajuće Orakl tipove.
- Automatsko rešavanje konflikata koji se tiču upotrebe rezervisanih Orakl reči.
- Čuvanje metapodataka u repozitorijumu gde se mogu vršiti izmene nad njima.
- Povraćaj informacija o izvornoj bazi kroz postupak obrade metapodataka izvorne baze.
- Parsiranje i transformacija pogleda u Orakl PL/SQL poglede.
- Obezbeđivanje mogućnosti izmena pri preslikavanju između tipova podataka, brisanju i preimenovanju objekata.
- Pravljenje izveštaja u vezi statusa poslova migracije.
- Pravljenje DDL skriptova za definisanje šeme ciljne Orakl baze.
- Pravljenje skriptova za prenos podataka.
- Dopuštanje da se migriraju objekti u već postojeće prostore za tabele.
- Prikaz grešaka i upozoravajućih poruka o poslovima migracije koji se izvršavaju.

U narednom delu će biti detaljno objašnjen proces migracije šeme DB2 baze korišćenjem Orakl SQL Developer alata za migraciju. Kao i za svaki alat, i za SQL Developer postoje određeni preduslovi u smislu poslova koje je potrebno odraditi kako bi se ovaj alat mogao koristiti za poslove migracije.

4.2 Prirprema okruženja za migraciju

Posedovanje osnovnog predznanja u vezi korišćenja Orakl SQL Developer alata može poslužiti kao dobra osnova pri upotrebi ovog alata od strane korisnika u vršenju poslova migracije. Skup poslova koje je potrebno udraditi kao preduslov, da bi se ovaj alat mogao koristiti za migraciju, varira u zavisnosti od prirode projekta migracije. U svakom slučaju može se prepoznati skup osnvnih poslova koje je potrebno udraditi da bi se okruženje pripremilo za proces migracije. U te poslove spadaju:

- Pravljenje naloga za korisnika repozitorijuma za migraciju
- Definisanje repozitorijuma za migraciju
- Registrovanje JDBC drajvera za pristup DB2 bazi
- Podešavanje ciljne šeme na Orakl bazi
- Podešavanje direktorijuma za projekat migracije

4.2.1 Pravljenje naloga za korisnika repozitorijuma za migraciju

SQL Developer zahteva postojanje repozitorijuma za migraciju u kome će se čuvati model metapodataka koji se dobije posle faze procesiranja izvorne DB2 baze. Ukoliko ne postoji ni jedan korisnik repozitorijuma onda je potrebno napraviti korisnika repozitorijuma čijoj šemi će biti pridružen repozitorijum.

Korisnik u čijoj se šemi nalazi repozitorijum potrebno je da ima minimum sledeće privilegije odnosno uloge:

- RESOURCE
- CREATE SESSION i CREATE VIEW

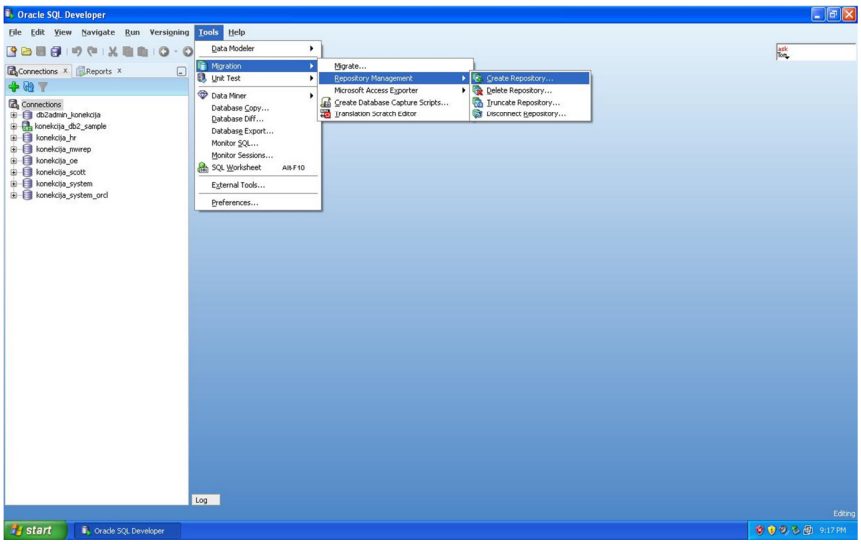
Ukoliko se simultano migrira više baza na Orakl platformu onda su potrebne dodatne uloge odnosno privilegije:

- RESOURCE ulozi mora biti dodeljena ADMIN opcija
- CREATE ROLE, CREATE USER i ALTER ANY TRIGGER ulozi mora biti dodeljena ADMIN opcija

Kada je definisan korisnik repozitorijuma, sledeći korak je definisanje konekcije u SQL Developer-u za tog korisnika.

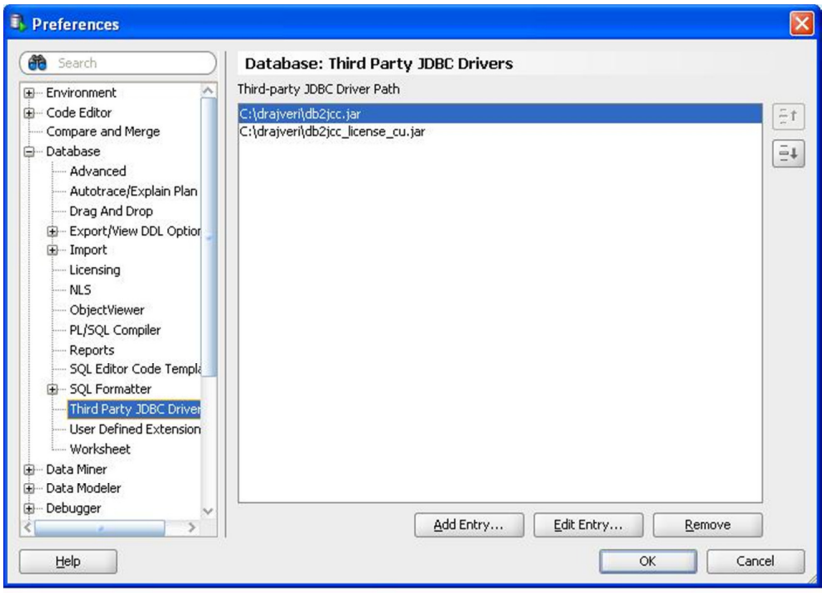
4.2.2 Definisanje repozitorijuma za migraciju

Kada je definisan korisnik repozitorijuma, sledeći korak je pridruživanje repozitorijuma šemi tog korisnika. Repozitorijum je pridružen jednom korisniku baze i kako SQL Developer dozvoljava pravljenje više repozitorijuma za potrebe različitih procesa migracija na Orakl bazu potrebno je svakom repozitorijumu pridružiti jednog odgovarajućeg korisnika. Ovaj pristup omogućava da se konkurentno vrše poslovi dva različita procesa migracije u odvojenim sesijama. Ukoliko nije bitno da se poslovi dva različita procesa migracije izvršavaju u istom vremenskom trenutku onda se samo jedan repozitorijum i njegov odgovarajući korisnik mogu koristiti za različite procese migracije koji se odvijaju jedan za drugim. Migracioni repozitorijum može biti pridružen odgovarajućem prethodno definisanom korisniku sa odgovarajućim dodeljenim ulogama i privilegijama tako što se izvrši desni klik tastera miša nad konekcijom korisnika repozitorijuma i odabere opcija Associate Migration Repository (Tools | Migration| Repository Management).



4.2.3 Registrovanje JDBC drajvera

Da bi se uspostavila konekcija iz SQL Developera prema nekoj ne-Orakl bazi potrebno je obezbediti specifični drajver za tu bazu. Za DB2 bazu odgovarajući drajver je dostupan na internetu arhiviran u .jar formatu i njegov naziv je *db2jcc.jar.* Ovaj specifični DB2 JDBC drajver može biti registrovan u SQL Developer-u pomoću opcije Third Party JDBC Drivers (Tools | Preferences | Database).



4.2.4 Pravljenje ciljne šeme na Orakl bazi

Pravljenje ciljne šeme(korisnika) na Orakl bazi, u kojoj će biti sačuvani migrirani objekti sa izvorne baze, može takođe biti jedan od preduslova koji mora biti ispunjen pri podešavanju okruženja za migraciju. Ovaj korak u podešavanju okruženja se može posmatrati kao opcioni. Ukoliko se pravi šema, odnosno korisnički nalog koji odgovara toj šemi, potrebno je da on poseduje minimum CONNECT/RESOURCE privilegiju kako bi napravio migrirane objekte na Orakl bazi. Tokom procesa migracije SQL Developer napravi skriptove za definisanje ciljnog korisnika(šeme) pošto se proces migracije završi. Taj skript je napravljen u formi DDL skripta.

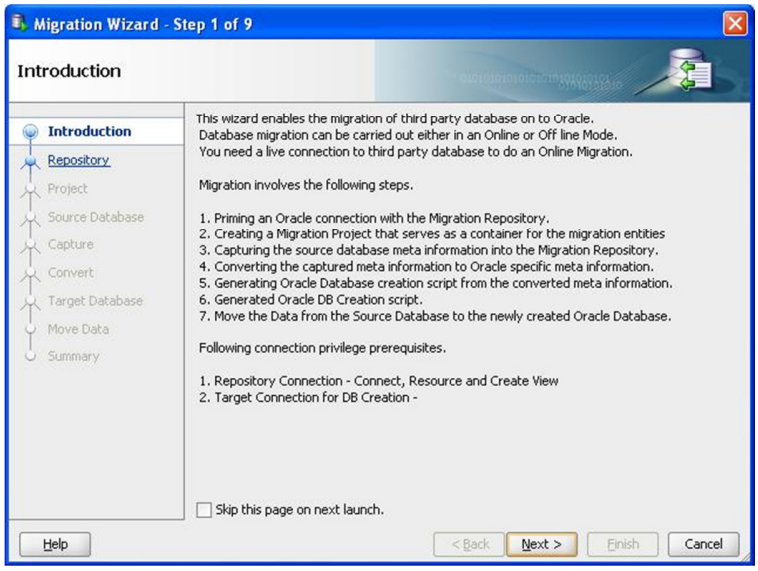
4.2.5 Definisanje direktorijuma za projekat migracije

Kao još jedan od uslova koji je potrebno da bude ispunjen pre samog početka procesa migracije, jeste i postojanje radnog direktorijuma, koji će služiti alatu za migraciju kao mesto na kome će čuvati sve skriptove i log datoteke koje napravi u toku procesa migracije. Jedan direktorijum može poslužiti za više projekata u kojima se SQL Developer koristi kao alat za migraciju.

Na kraju, kada su svi neophodni uslovi ispunjeni, započinje se proces migracije šeme DB2 baze podataka.

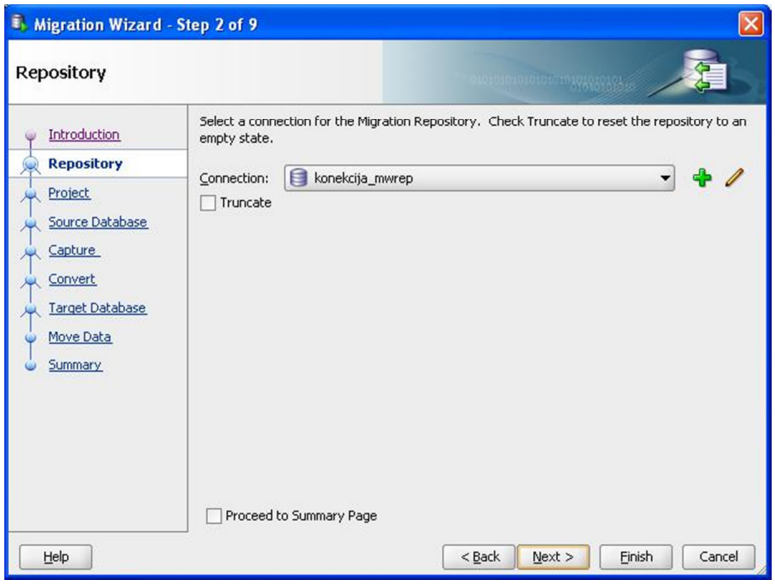
4.3 Migracija šeme DB2 baze

Migracija šeme baze uz pomoć SQL Developer alata predstavlja posao koji se sprovodi u sedam koraka uz pomoć čarobnjaka za migraciju obezbeđenog od strane alata. Čarobnjak za migraciju može biti pokrenut tako što se izvrši desni klik mišem na konekciju za izvornu bazu i izabere opcija Migrate to Orakl ili iz glavne palete sa alatkama pomoću opcije Migrate (Tools | Migration | Migrate) posle čega se pojavi početni prozor koji je prikazan na sledećoj slici.



4.3.1 Odabir konekcije za repozitorijum

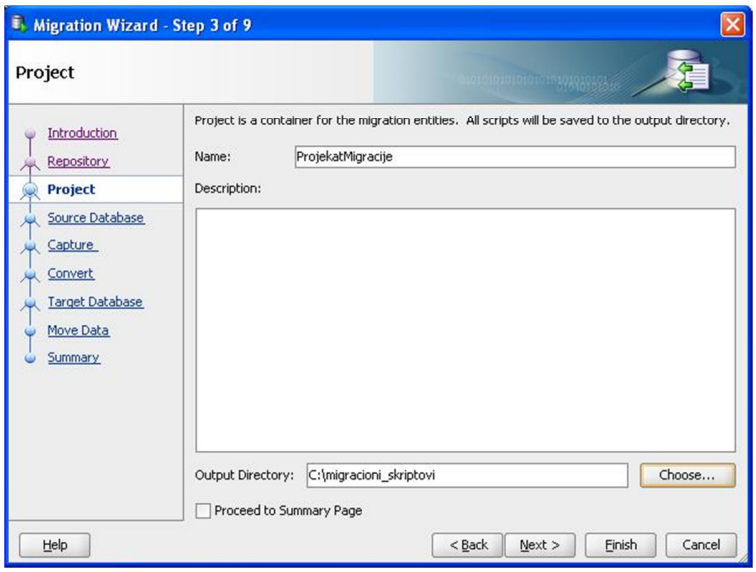
U prvom koraku se vrši odabir konekcije za migracioni repozitorijum. Kao što je prethodno rečeno, moguće je imati više migracionih repozitorijuma tako da je podržano konkurentno migriranje više različitih baza na Orakl RSUBP. U takvim uslovima potrebno je naznačiti željeni migracioni repozitorijum u kome će se čuvati sve informacije prikupljene od strane alata posle izvršavanja svakog od koraka tokom procesa migracije.



4.3.2 Definisanje projekta migracije

U ovom koraku je potrebno navesti naziv projekta migracije i ukazati na radni direktorijum koji je prethodno napravljen na fizičkom disku. U radnom direktorijumu se čuvaju sve specifične informacije koje se tiču samog projekta migracije kao što su informacije o modelu metapodataka izvorne baze, informacije o konvertovanom modelu, različiti skriptovi koji su napravljeni tokom procesa

migracije.Takođe i DDL skriptovi za pravljenje ciljne Orakl šeme, koji su na kraju napravljeni, sačuvani su u ovom direktorijumu.



4.3.3 Prikupljanje metapodataka izvorne baze

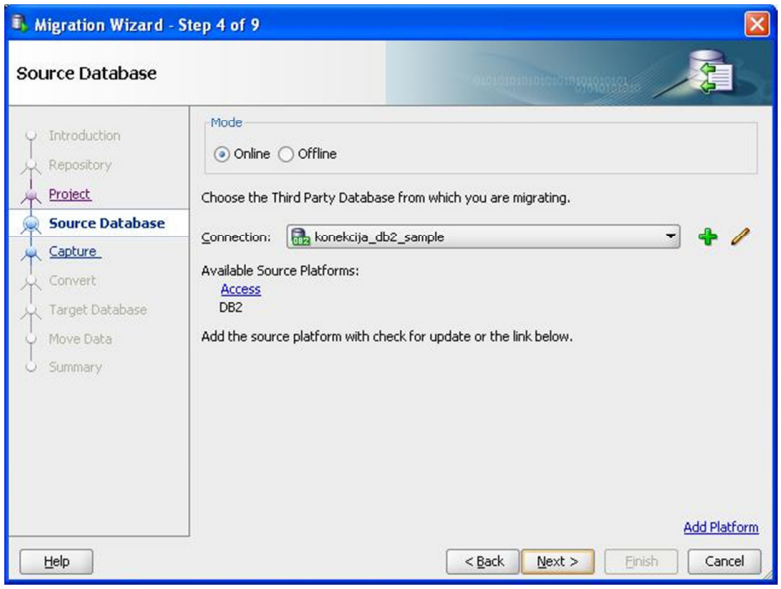
Da bi izvršio poslove migracije baze odnosno šeme, SQL Developer prvo prikuplja metapodatke izvorne baze zahvaljujući kojim vrši konverziju i sprovodi analizu izvorne baze koja pomaže pri planiranju poslova projekta migracije. Takođe se prikupljeni metapodaci u repozitorijumu mogu koristiti prilikom pravljenja relevantnog izveštaja za svaku izvršenu fazu migracije.

Kada su metapodaci o objektima izvorne baze dostupni u repozitorijumu za migraciju, SQL Developer omogućava pretraživanje, uklanjanje objekata sa greškom i odstranjivanje onih koji su nepoželjni u migriranoj bazi.

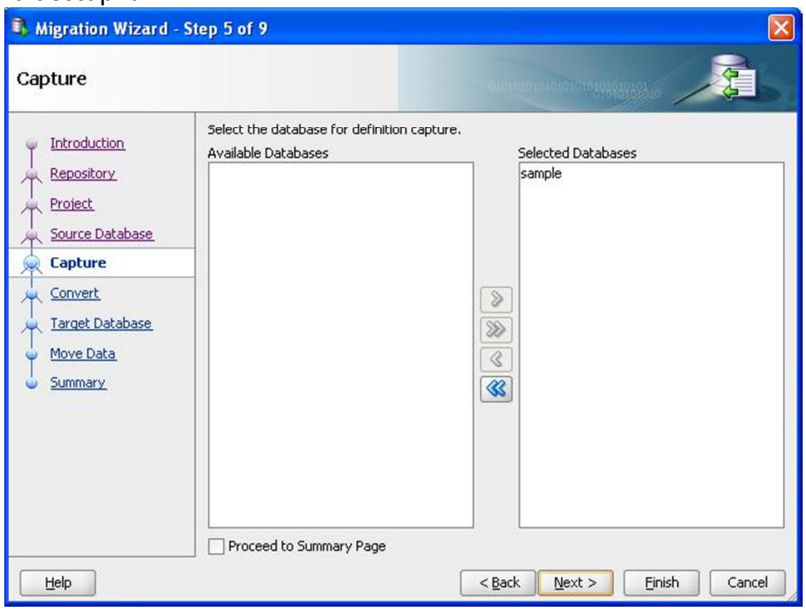
SQL Developer dozvoljava prikupljanje metapodataka izvorne baze na dva načina – *offline* i *online*. Za velike baze koje sadrže više hiljada objekata, preporučen je *offline* način prikupljanja metapodataka izvorne baze čime se rizik svodi na minimum. Za slučaj manjih baza preporučen je *online* način prikupljanja metapodataka.

4.3.3.1 *Online* način prikupljanja metapodataka izvorne baze

Kao i što samo ime sugeriše, *online* proces prikupljanja metapodataka izvorne baze podrazumeva uspostavljenu konekciju ka izvornoj bazi koristeći odgovarajući JDBC drajver. Da bi prikupio sve potrebne metapodatke, potrebno je da korisnik preko koga je uspostavljena konekcija ima administratorske privilegije na izvornoj DB2 bazi. Pri oba načina, SQL Developer omogućava opciju izbora konekcije ka izvornoj bazi radi obrade njenih metapdataka.

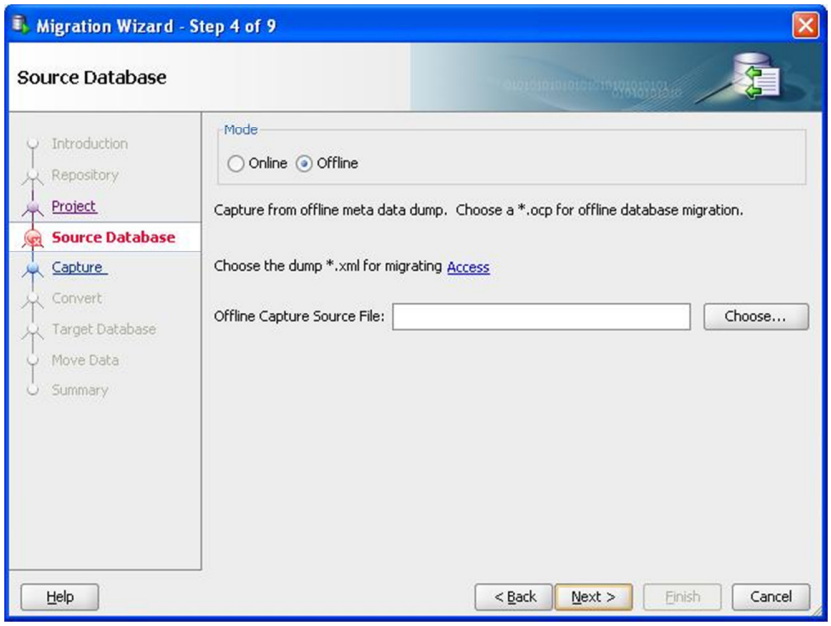


Posle odabira konekcije, migracioni čarobnjak će pokrenuti prozor koji će izlistati sve dostupne baze omogućavajući korisniku da odabere željenu bazu čiji će se metapodaci obraditi. U ovom koraku je moguće odabrati više željenih baza za migraciju istovremeno ili je automatski odabrana jedna baza ukoliko je samo ona dostupna.



4.3.3.2 *Offline* način prikupljanja metapodataka izvorne baze

Druga opcija koju omogućava SQL Developer prilikom započinjanja procesa prikupljanja i obrade metapodataka izvorne baze jeste *offline* mod u kome će se ovaj proces izvršiti. Odabirom ove opcije dobija se prozor kao na sledećoj slici.

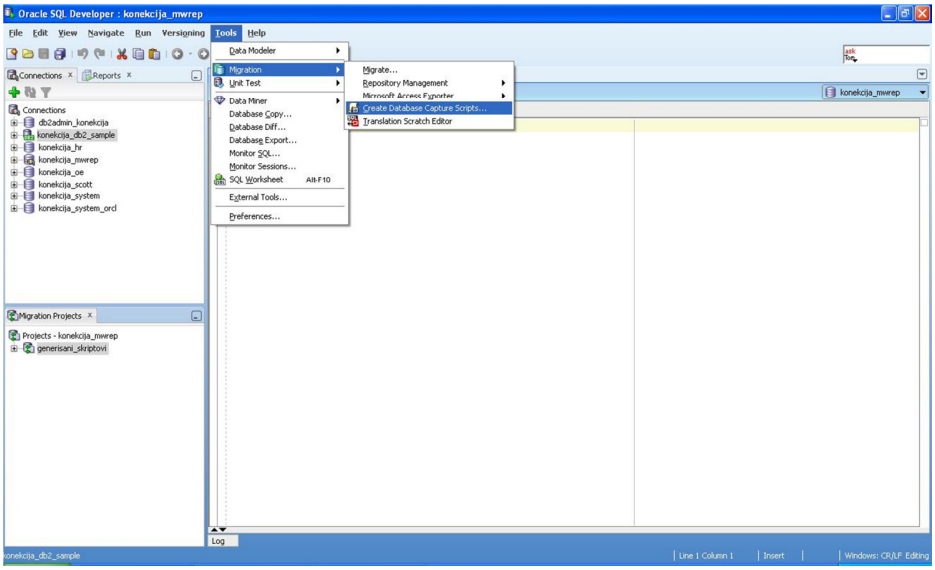


Da bi se nastavilo dalje potrebno je naznačiti putanje do nestruktuiranih datoteka(flat fajlovi) u kojima su svi metapodaci izvorne baze sačuvani.

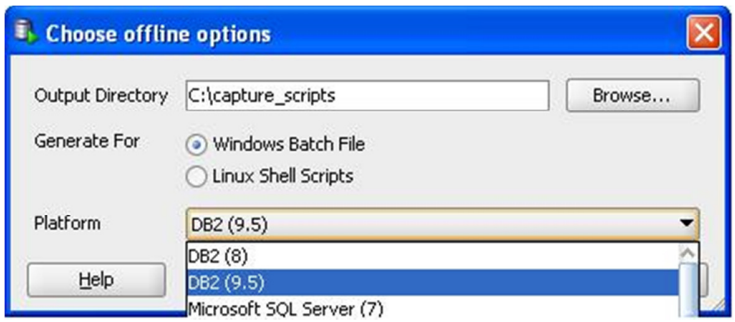
Za ovaj način prikupljanja metapodataka izvorne baze, karakteristično je ručno izvršavanje niza koraka kako bi se prikupili metapodaci izvorne baze u nestruktuirane datoteke a potom iz njih učitali u migracioni repozitorijum za dalje akcije u procesu migracije. U ovaj proces su uključeni sledeći koraci:

1. Uz pomoć SQL Developer-a se najpre naprave skriptovi, koji su karakteristični za DB2 bazu, i čija je uloga da prikupe metapodatke izvorne baze u nestruktuirane datoteke. Prilikom pravljenja ovih skriptova od strane alata moguće je odabrati opciju koja ukazuje da li se skriptovi prave za Windows platformu (BAT datoteka) ili za Linux/Unix platforme (Shell skriptovi). Takođe je u okviru ovog koraka potrebno podesiti direktorijum u kome će biti sačuvani skriptovi napravljeni od strane alata. Za ove potrebe može biti iskorišćen radni direktorijum projekta migracije s tim što je preporučeno da se skriptovi za prikupljanje metapodataka izvorne baze čuvaju u zasebnom poddirektorijumu kako ne bi došlo do njihovog mešanja sa ostalim skriptovima napravljenim tokom procesa migracije.

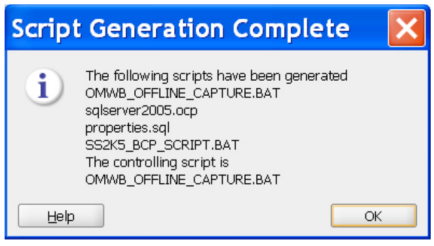
2. Da bi se napravili ovi skriptovi potrebno je iz SQL Developer-a pokrenuti opciju Create Database Capture Scripts (Tools |Migration | Create Database Capture Scripts) kao što je prikazano na sledećoj slici.



3. Potom je potrebno odabrati direktorijum u kome će biti sačuvani napravljeni skriptovi i odabrati tip baze, u ovom slučaju DB2, koju migriramo na Orakl RSUBP. Ovu akciju vršimo na način koji je prikazan na sledećoj slici.



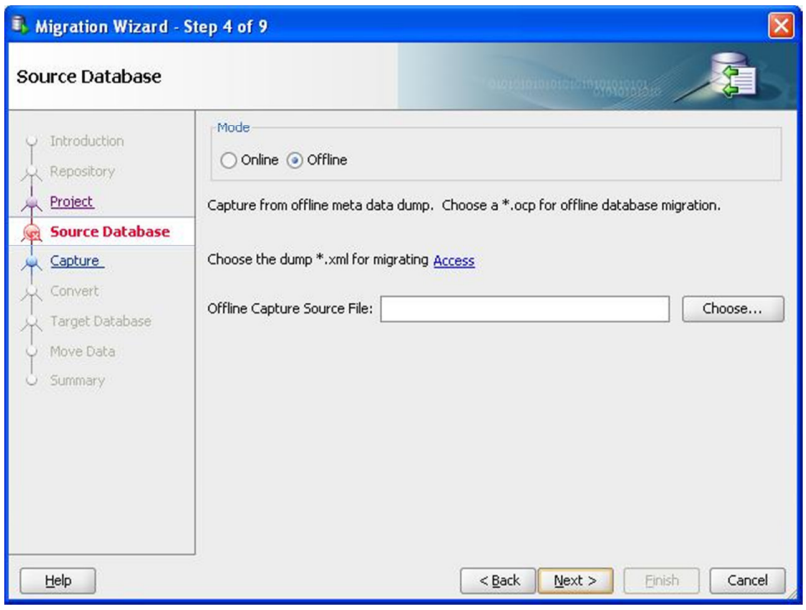
4. Skriptovi, pomoću kojih će se učitati metapodaci izvorne DB2 baze u nestruktuirane datoteke sačuvani su u direktorijumu koji je naznačen u prethodnom koraku.



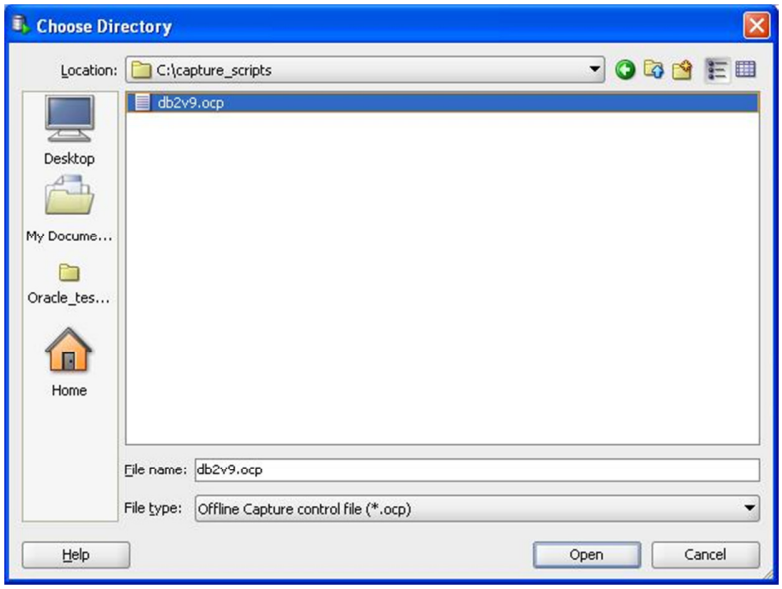
5. Pošto su skriptovi napravljeni, potrebno ih je prebaciti na server na kome se nalazi izvorna DB2 baza i izvršiti ih sa odgovarajućim parametrima koji su specifični za svaku bazu. Ovi parametri su korisničko ime koje imas privilegije administratora, njegova lozinka i naziv baze. Preporučeno je da svi skriptovi koji obrađuju metapodatke izvorne baze budu sačuvani u jednom zasebnom direktorijumu odakle ih je lako izvršiti.

6. Posle završetka izvršavanja skriptova, sve datoteke u direktorijumu u kome su skriptovi sačuvani uključujući i nove napravljene datoteke sa metapodacima izvorne baze i nove poddirektorijume potrebno je prebaciti na računar na kome je instaliran SQL Developer.

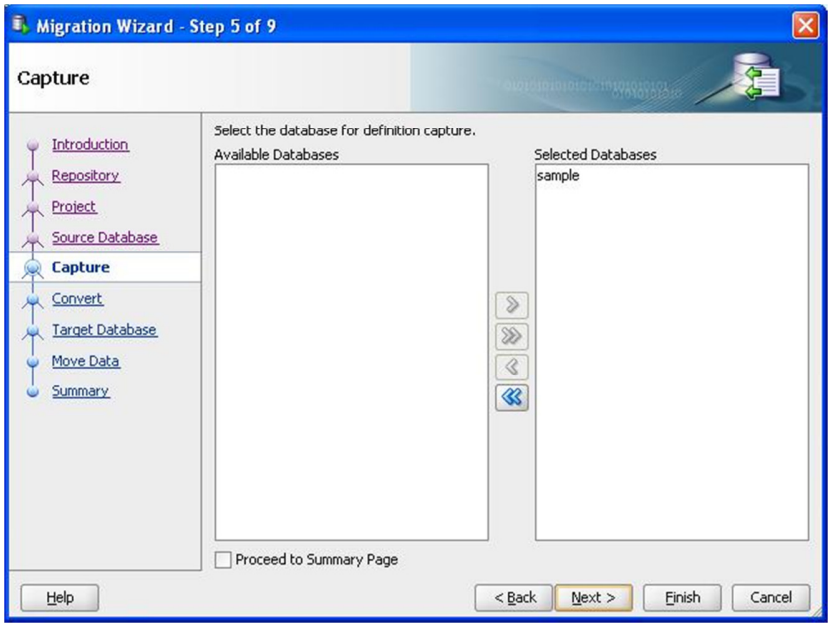
7. Kada su datoteke u kojima su sačuvani metapodaci izvorne baze dostupni SQL Developer-u, proces učitavanja metapodataka iz njih u migracioni repozitorijum može da se započne korišćenjem *offline* opcije u migracionom čarobnjaku tokom faze prikupljanja metapodataka izvorne baze.



8. Pošto je izabran *offline* način prikupljanja metapodataka izvorne baze, potrebno je da korisnik ukaže na direktorijum u kome su datoteke sa metapodacima izvorne baze sačuvani. Glavna datoteka na koju treba ukazati u ovom koraku ima .ocp ekstenziju.



9. U sledećem koraku potrebno je odabrati izvornu bazu koja se migrira na Orakl RSUBP iz liste svih baza koje su dostupne. Ovaj korak je važan, jer skriptovi za obradu metapodataka izvorne baze u nestruktuiranim datotekama mogu da sačuvaju metapodatke svih baza na izvornom DB2 sistemu za upravljanje bazom podataka. Korisnik odabirom jedne ili više željenih izvornih baza ukazuje alatu čije metapodatke je potrebno spremiti u repozitorijum za migraciju.



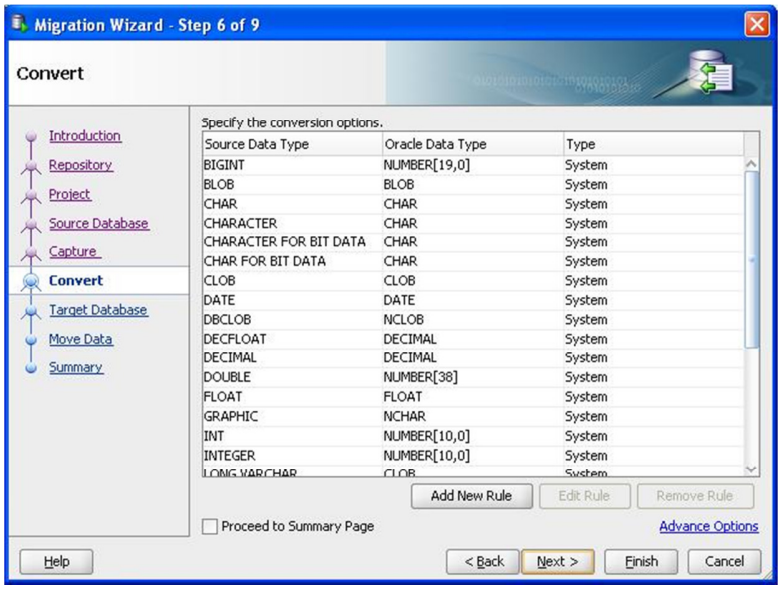
10. Posle završetka ovog koraka, proces učitavanja metapodataka u migracioni repozitorijum još nije krenuo sa izvršavanjem. Ovaj proces će početi pošto se izvrše svi sledeći koraci u migracionom čarobnjaku. Ukoliko želimo da odmah započnemo proces učitavanja metapodataka u migracioni repozitorijum potrebno je odabrati opciju *Proceed to Summary Page*.

Ovim poslednjim izvršenim korakom metapodaci izvorne baze su spremljeni u migracionom repozitorijumu i uz pomoć njih je napravljen takozvani *captured* model izvorne baze. Sledeća faza u procesu migracije jeste konverzija ovog modela u Orakl model.

4.3.4 Konverzija modela izvorne DB2 baze u Orakl model

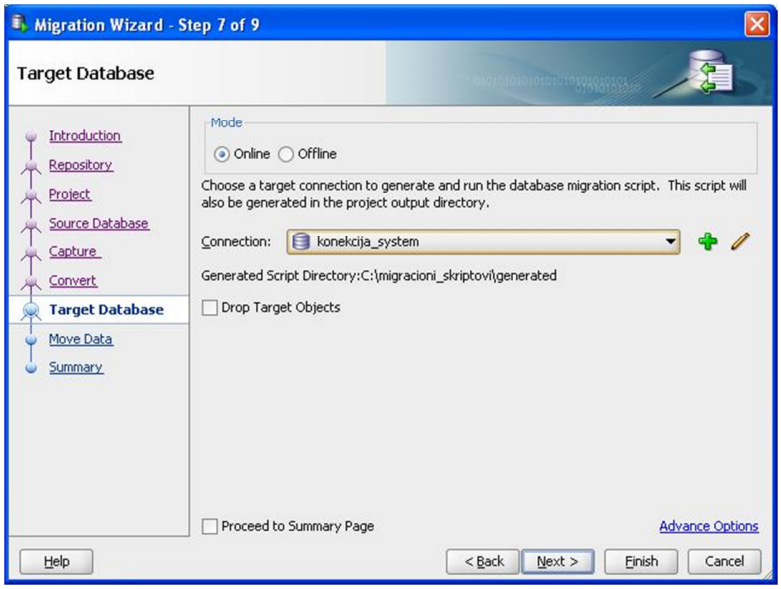
U fazi konverzije modela, kao što samo ime sugeriše, konvertuje se model izvorne DB2 baze u ekivalentni Orakl model. Posle završetka ove faze u migracionom repozitorijumu se nalaze i model preseka stanja izvorne DB2 baze i njemu odgovarajući konvertovani Orakl model. Tokom procesa konverzije, SQL Developer napravi šemu i njene objekte koji su odgovarajući za Orakl bazu. U ovoj fazi projekta migracije, odgovarajući logički model za buduću Orakl šemu je napravljen i sačuvan u migracionom repozitorijumu.

Tokom faze konverzije, SQL Developer korisniku prikazuje listu svih tipova korišćenih u izvornoj DB2 bazi i njima odgovarajućih Orakl tipova u koje će se oni preslikati. Omogućeno je korisniku, koji vrši migraciju, da menja podrazumevana pravila preslikavanja tako što specifikuje odgovarajuće tipove u Orakl bazi. Najčešći tipovi podataka koji se preslikavaju i na koje treba obratiti posebnu pažnju su: CHAR, VARCHAR, TIMESTAMP, DATE, CLOB/ BLOB. Obično svi numerički tipovi su preslikani u NUMBER tip u Orakl bazi. Jednostavni korisnički tipovi su konvertovani u njima odgovarajuće osnovne Orakl tipove, dok neke složenije korisnički definisane tipove nije moguće na ovaj način konvertovati zbog ograničenja alata za migraciju.



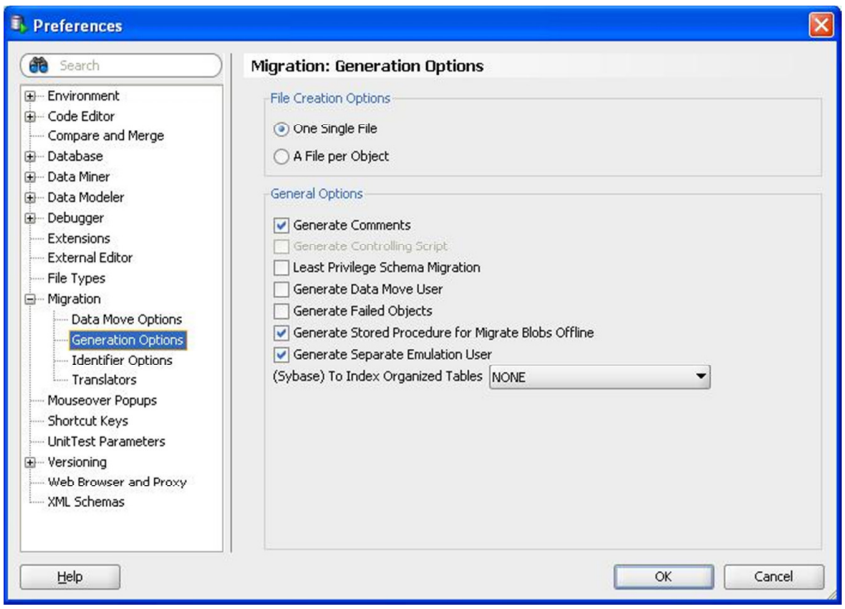
4.3.5 Pravljenje šeme Orakl baze

U sledećem koraku procesa migracije je potrebno definisati gde i na koji način će ciljna Orakl šema biti napravljena. Ukoliko je odabran pristup pravljenja *online* načinom, SQL Developer pravi Orakl šemu koristeći konekciju Orakl korisnika koji poseduje privilegiju definisanja šeme(korisnika) u bazi. Ukoliko je takva korisnička konekcija napravljena kao deo poslova koje je potrebno obaviti u cilju pripreme okruženja za migraciju, onda ona može biti iskorišćena u ovom koraku.



Ukoliko korisnik sa privilegijom pravljenja šeme nije napravljen u sklopu poslova pripreme okruženja za migraciju, SQL developer omogućava da se taj posao izvrši u ovom koraku pre izbora same konekcije. Šema koja odgovara toj konekciji nije ciljna šema, već njen korisnik zahvaljujući privilegijama koje su mu dodeljene je u mogućnosti da napravi šemu u kojoj će biti sačuvani migrirani objekti iz izvorne DB2 baze.

Ukoliko je odabran *offline* pristup prilikom pravljenja ciljne šeme Orakl baze, SQL Developer omogućava pravljenje jednog velikog skripta za pravljenje ciljne šeme i objekata u njoj ili pravljenje više skriptova odvojeno za svaki objekat koji migrira na Orakl bazu. Ova podešavanja su moguća zahvaljujući linku Advanced Option koji se nalazi u donjem levom uglu prethodnog prozora migracionog čarobnjaka. Svi skriptovi napravljeni u ovom koraku su sačuvani u radnom direktorijumu projekta migracije. Oni mogu biti izvršeni kasnije, pri čemu se mogu i izmeniti ukoliko postoji potreba za izmenama. Dodatna podešavanja koja je moguće odraditi u ovom koraku predstavljena su na sledećoj slici.



Ukoliko je naznačena Generate Failed Objects opcija, SQL Developer će uključiti i DDL za objekte koje nije konvertovao prilikom pravljenja skriptova za pravljenje šeme, što omogućuje korisniku koji vrši migraciju da identifikuje sve objekte tog tipa i pronađe razlog zbog čega nisu prošli proces konverzije.

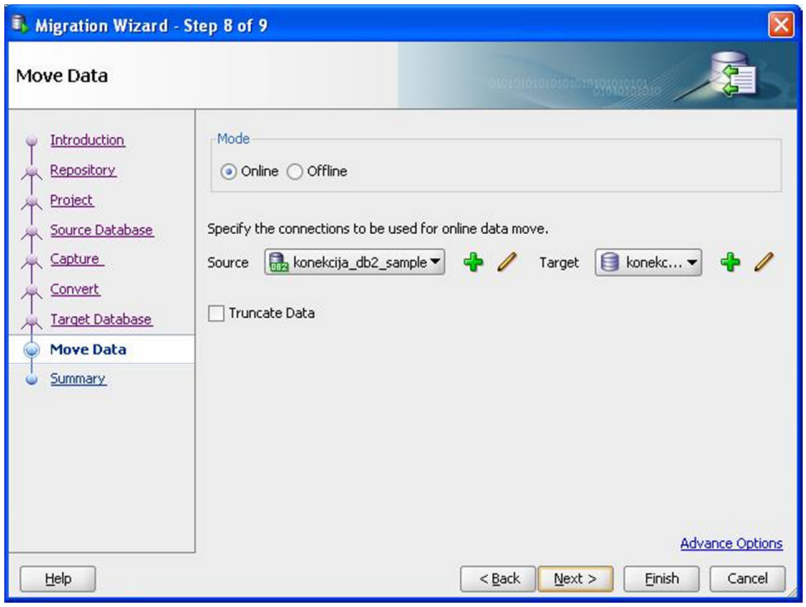
4.3.6 Migracija podataka na Orakl bazu

SQL Developer olakšava proces migracije podataka iz izvorne DB2 baze u ciljnu Orakl bazu pristupom na dva načina. Ukoliko je u pitanju *online* mod, SQL Developer koristi JDBC konekciju za pristup i izdvajanje podataka iz izvorne DB2 baze. Ukoliko je u pitanju *offline* mod, SQL Developer napravi skriptove za izdvajanje podataka iz izvorne DB2 baze u nestruktuirane datoteke i uz pomoć SQL Loader-a, Orakl alata za učitavanje podataka u bazu, učitava podatke u ciljnu Orakl bazu. Za pristup migraciji podataka u *offline* modu, SQL Developer napravi sve potrebne skriptove za izdvajanje i učitavanje podataka, skriptove pomoću kojih se uključuju odnosno isključuju ograničenja i kontrolne skriptove koji pokreću skriptove za izdvajanje i učitavanje podataka koji se koriste kako bi se automatizovali i smanjili administratorski poslovi pri obavljanju posla migracije podataka.

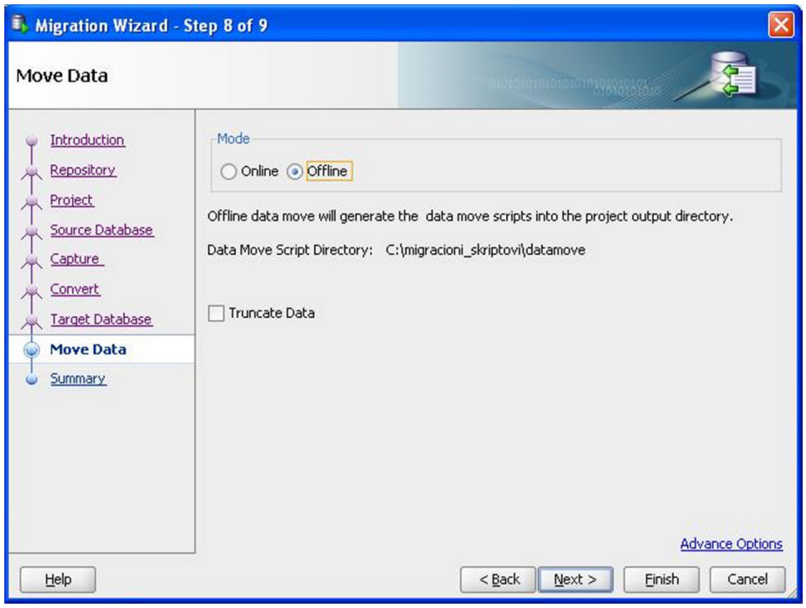
Online način pri migraciji podataka je veoma koristan pristup i on je pogodan iz nekoliko sledećih razloga:

- Dozvoljava da se upotrebi određeni broj već postojećih konekcija za migraciju podataka sa izvorne DB2 baze na Orakl bazu. Ova opcija omogućava optimizaciju procesa migracije podataka koji je baziran na dostupnosti mreže između izvornog i cilljnog RSUBP-a.
- Ponovne upotrebe JDBC konekcija, ka izvornoj i ciljnoj bazi, koje su napravljene u prethodnim fazama migracije. Jedino je potrebno da korisnik odabre konekcije čijom upotrebom olakšava proces migracije podataka bez obzira na broj tabela u izvornoj odnosno ciljnoj šemi.
- Dozvoljava brzo brisanje podataka iz ciljne šeme, ukoliko je se u toku procesa migracije podataka desila greška pa je potrebno čitav proces pokrenuti iz početka.

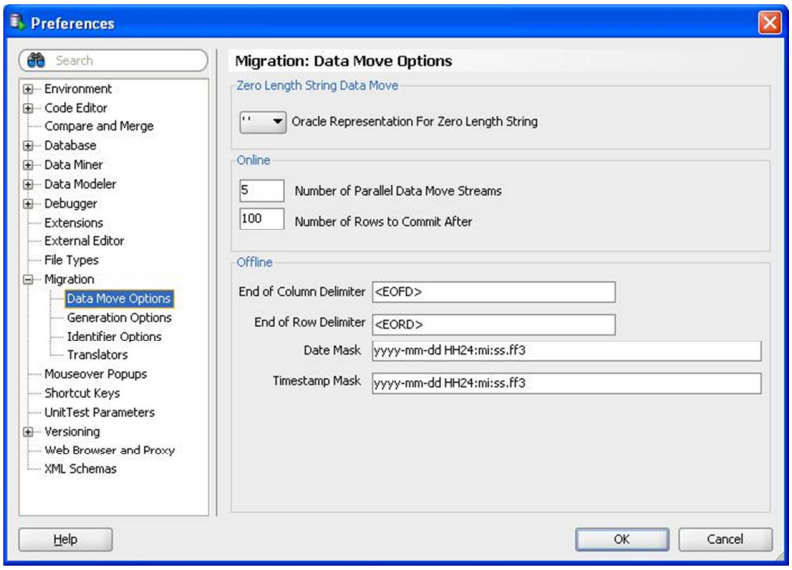
Za male i srednje baze *online* mod je preporučen pristup pri migraciji podataka.



Ukoliko se koristi *offline* mod migracije podataka, SQL Developer napravi skriptove za izdvajanje podataka koji koriste alate za izdvajanje podataka DB2 izvorne baze kao što je UNLOAD komanda.



SQL Developer dozvoljava korisniku da ručno prilagodi formate vremenskih podataka iz izvorne DB2 baze, naznake za kraj reda i separatore u datotekama koje sadrže podatke iz izvorne baze. Ova dodatna podešavanja su omogućena zahvaljujući Advanced Option linku koji otvara sledeći prozor.



Na kraju procesa migracije, migracioni čarobnjak otvara prozor u kome daje pregled svih opcija koje su odabrane od strane korisnika tokom procesa migracije a odnose se na fazu obrade metapodataka izvorne baze, fazu konverzije, pravljenja ciljne Orakl šeme i migraciju podataka. Ukoliko su sva podešavanja u redu, migracioni proces može da počne pri čemu će se koraci u njemu izvršavati prema izboru koji je napravljen od strane korisnika tokom procesa migracije.

Ukoliko je proces migracije uspešno završen, vrši se provera objekata napravljenih u Orakl bazi sa objektima u izvornoj DB2 bazi. Ova provera podrazumeva broj objekata i njihov status posle migriranja. Veoma je važno proveriti migracione logove i izveštaje koje SQL Developer pravi po završetku svake od faza migracije. Na osnovu izveštaja u njima moguće je ustanoviti za koje objekte je se javila greška prilikom migracije, kao i koji objekti nisu u potpunosti migrirali zbog određenih ograničenja u samom alatu za migraciju. U izveštajima i log datotekama SQL Developer prikazuje sve moguće informacije koje se tiču objekata koji su procesirani iz izvorne baze, konvertovanih objekata, napravljenih objekata u ciljnoj Orakl bazi, njihovog konačnog statusa i problema koji su se pojavili prilikom migracije ovih objekata.

Kao što se može videti iz prethodnog opisa, poslovi migracije šeme i podataka DB2 baze se mogu izvršiti efikasno uz pomoć Orakl alata za migraciju SQL Developer-a koristeći veoma intuitivan čarobnjak za migraciju i čitajući izveštaje koje alat napravi prilikom izvršavanja poslova migracije. Izveštaji takođe pomažu pri verifikaciji prethodno podešenih parametara i pravljenju procena napora i vremena koje je potrebno utrošiti na obavljanje poslova migracije.

4.4 Migracija tipova podataka

Svaka vrednost koja se čuva u bazi podataka ima svoj tip podatka. Kad govorimo o tipu podatka koji odgovara nekoj vrednosti onda mislimo na određen skup svojstava te vrednosti. Ovaj skup svojstava ukazuje sistemu za upravljanje bazom podataka na koji način da tretira vrednosti koje odgovaraju jednom tipu podatka u odnosu na vrednosti koje odgovaraju nekom drugom tipu podatka. Tako na primer, vrednosti tipa NUMBER se mogu sabirati, dok se vrednosti tipa CHAR ne mogu sabirati.

Prilikom definisanja tabele za svaku njenu kolonu mora biti naznačen tip podatka. Takođe prilikom definisanja funkcija i procedura tip podatka mora biti naznačen za njihove argumente. Tip podatka naznačava domen vrednosti koje svaka kolona ili argument mogu imati. Sistemi za upravljanje bazom podataka obezbeđuju ugrađene tipove kao i mogućnost definisanje tipova podataka od strane korisnika.

Uspešnost konverzije tipova podataka između različitih sistema za upravljanje bazom podataka je od presudnog značaja u procesu migracije. Podaci su svakako najvažniji kada govorimo o bazi uopšte i ukoliko njihovi tipovi ne migriraju na pravi način, u smislu njihove semantike, onda se može pojaviti dosta problema sa radom migriranih aplikacija na ciljnom sistemu za upravljanje bazom podataka. Posebnu pažnju treba obratiti na nestandardizovane tipove koji se javljaju kod pojedinih sistema za upravljanje bazom podataka kao što je Orakl. Tu razlikujemo osnovne tipove kao što su NUMERIC, STRING i DATE i složene tipove dostupne u PL/SQL programskom jeziku a to su slogovi i kolekcije.

4.4.1 Preslikavanje DB2 tipova podataka u Orakl tipove

U ovom delu će biti predstavljena podrška od strane relacionog sistema Orakl za najvažnije i najčešće korišćene DB2 tipove.

NUMBER tip podatka u Orakl bazi daje podršku za celobrojne numeričke tipove iz DB2 baze kao što su BIGINT, INTEGER, SMALLINT. U Orakl bazi je moguće definisati tabele čijim kolonama odgovaraju svi prethodno navedeni DB2 numerički tipovi, s tim što se vrši njihova implicitna konverzija u Orakl NUMBER tip. S druge strane, Orakl tipu REAL za realne numeričke vrednosti odgovaraju DB2 tipovi FLOAT, DECIMAL, DOUBLE, REAL.

Što se tiče karakter tipova, tu postoji znatna razlika između relacionih sistema Orakl i DB2 koja se ogleda u razlici količine podataka koju je dozvoljeno smestiti u promenljive ovog tipa. U Orakl bazi je odvojeno najviše 4000 bajtova za karakter tipove CHAR i VARCHAR, dok je u DB2 bazi za ove tipove odvojeno 32 767 bajtova. Prilikom migracije ove razlike se prevazilaze tako što DB2 karakter tipovi čija je dužina veća od 4000 bajtova iziskuju upotrebu CLOB Orakl tipova prilikom njihove konverzije. Ovaj postupak će SQL Developer udraditi implicitno prilikom konverzije karakter tipova. Kasnija upotreba CLOB objekata umesto jednostavnih DB2 karakter tipova zahteva sve potrebne izmene u kodu aplikacija kako bi one nastavile da rade u novom okruženju. Javlja se potreba za korišćenjem LOB API-ja u kodu aplikacija umesto direktnog pristupa karakter podacima.

Uopšteno gledajući može se primetiti različit nivo podrške za vremenske podatke između različitih sistema za upravljanje bazom podataka. Neke baze imaju ugrađenu podršku za TIME tip podatka u kome se čuva informacija o vremenu u toku jednog dana, dok druge imaju jedinstven tip za čuvanje podatka o vremenu u toku dana i dodatni podatak koji govori o tome koji je dan u godini. Relacione baze se razlikuju i po granularnosti vremenskog podatka koji je podržan od strane ovih tipova. Razlike između RSUBP Orakl i DB2 u smislu podrške za TIME i DATE/TIMESTAMP tipove podataka se ogledaju u sledećem:

- U Orakl bazi postoji podrška za vremenski tip DATE u kome se čuva zajednički podatak o datumu i vremenu. S druge strane ne postoji podrška za tip TIME u kome se čuva podatak o trenutnom vremenu u toku dana.
- U DB2 bazi postoji podrška za TIME tip podatka i za DATE tip podatka u kome se čuva samo informacija o datumu bez dodatne informacije o vremenu.
- Podrška za DateTime tip podatka postoji u oba RSUBP-a i ona je implementirana preko TIMESTAMP tipa podatka, pri čemu je preciznost za ovaj tip u Orakl bazi izražena u mikrosekudama dok je u DB2 bazi ona izražena u milisekundama.

Migracija korisnički definisanih tipova sa DB2 baze na Orakl bazu je veoma složen i težak posao. SQL Developer konvertuje osnovne korisnički definisane tipove koji su bazirani na ugrađenim osnovnim Orakl tipovima kao što su CHAR, VARCHAR itd.

4.5 Opšti pregled migracije podataka između različitih RSUBP-ova

Migracija podataka je složen proces, sastavljen od više aktivnosti, kojim se iz starog RSUBP-a podaci prenose u novi sistem, pri čemu se nad podacima vrše sve potrebne transformacije prema zahtevima novog sistema.

Sam po sebi krajnji proces učitavanja podataka u ciljnu bazu je veoma brz, međutim pre početka tog procesa potrebno je vreme za organizaciju svih pripremnih poslova kako bi učitavanje podataka moglo da počne. Učitavanje podataka u bazu može zavisiti od mnogo faktora kao što su: konfiguracija servera na kome radi baza, konfiguracija I/O podsistema na serveru, podešenosti šeme u koju se migriraju podaci u smislu tipova podataka koji su korišćeni, broja indeksa koji su definisani nad tabelama, veličina kolona i redova podataka itd. Ukoliko se radi o migraciji velikih količina podataka onda taj proces može trajati i nekoliko nedelja. Ovaj proces zahteva precizno planiranje i pažljivo razvijenu metodologiju migracije podataka kako bi sam proces migracije bio što efikasniji.

Najčešće aktivnosti koje su deo procesa migracije između različitih RSUBP-ova podataka su:

Formulisanje strategije i plana migracije podataka - Definisanje jasne strategije i plana kako bi se postigao što veći nivo automatizacije poslova što bi pomoglo njihovo izvršavanje u predviđenom vremenskom roku. Potrebno je razmotriti sve ponuđene opcije, u smislu korišćenja softverskih alata, koje nude proizvođači RSUBP-ova za izdvajanje, prenos i učitavanje podataka.

Izdvajanje podataka iz izvorne baze - Postoji nekoliko opcija za ekstrakciju podataka iz izvorne baze i sve uključuju alate kao što su DB2 UNLOAD alat, Orakl Data Pump, ETL (Extract Transform Load) alat Orakl Data Integrator. Glavni posao posle ekstrakcije podataka iz izvorne baze jeste organizacija datoteka u kojima se nalaze podaci iz izvorne baze. On podrazumeva postojanje posebnog diska (“staging area”) na kome će se čuvati datoteke. Takođe se prilikom transfera velikih količina podataka pomoću FTP-a kroz mrežu, u kojoj se nalaze serveri na kojima rade baze, mora obratiti pažnja na sve nedostatke koje ovakav pristup proizvodi.

Verifikacija podataka - Verifikacija podataka predstavlja veoma važan posao u procesu migracije podataka. Za ovaj posao najčešće se razvijaju skriptovi koji verifikuju podatke posle njihovog učitavanja u ciljnu bazu. Takođe postoje određeni alati koji mogu poslužiti za profilisanje i proveru kvaliteta podataka. Prilkom provere kvaliteta podataka najčešće treba obratitit pažnju na sledeće stvari:

- Tačnost polja koja sadrže vremenske podatke. Nepravilna upotreba formata vremenskih podataka može uzrokovati gubljenje određenih informacija.
- Nepravilno određivanje veličine kolona. Posledica nepravilno određene veličine kolone za određeni tip podataka može skratiti ili zaokružiti određeni numerički podatak čime je narušen kvalitet tog podatka u smislu njegove preciznosti.
- Kolone koje sadrže karakter podatke. Ove kolone mogu sadržati takozvane otpadne podatke koji se javljaju kao posledica razlika u kodiranju podataka ovog tipa u izvornom i ciljnom RSUBP-u.

4.5.1 Učitavanje podataka u Orakl bazu

Najčešće korišćeni alat za učitavanje podataka u Orakl bazu jeste SQL Loader. Njegove najvažnije prednosti u odnosu na druge alate jesu to što je besplatan, fleksibilan i veoma brz. SQL Loader može da učita podatke iz datoteka i podržava više različitih formata podataka kao i različite pristupe pri kodiranju podataka. Pomoću ovog alata moguće je izvršiti učitavanje podataka u paketnoj obradi tako da se potvrđivanje učitanih podataka vrši češće čime se poboljšavaju performanse.

SQL Loader zahteva postojanje kontrolne datoteke koja u sebi sadrži sve zahtevane direktive za učitavanje podataka u Orakl bazu. Ovaj alat takođe daje mogućnost vršenja dodatnih operacija nad podacima pre njihovog učitavanja u bazu. Takođe je moguće podatke učitati u više različitih tabela iz jedne datoteke koja može sadržati slogove različite veličine. Ova mogućnost se retko koristi jer se podaci prilikom migracije čitave baze najčešće direktno prenose iz jedne tabele u drugu odgovarajuću tabelu u novoj bazi.

Drugi mogući pristup prilikom učitavanja podataka u Orakl bazu jeste preko eksternih tabela. Pomoću eksternih tabela je moguće učitati podatke iz datoteke u Orakl bazu. Za razliku od SQL Loadera koji predstavlja alat koji se koristi iz komandne linije, eksterne tabele se mogu koristiti u SQL naredbama. Zahvaljujući eksternim tabelama moguće je čitati podatke iz datoteke na isti način na koji se čitaju podaci iz neke tabele u bazi. Eksterne tabele su definisane kao i regularne tabele u bazi i zahtevaju informacije o lokaciji datoteke koja sadrži podatke, formatu podataka, separatorima kolona i redova podataka u datoteci itd. Sledi primer definicije jedne eksterne tabele:

```
CREATE TABLE emp_load
          (employee_number CHAR(5),
           employee_dob CHAR(20),
           employee_last_name CHAR(20),
           employee_first_name CHAR(15),
           employee_middle_name CHAR(15),
           employee_hire_date DATE)
   ORGANIZATION EXTERNAL
    (TYPE ORACLE_LOADER
     DEFAULT DIRECTORY def_dir1
     ACCESS PARAMETERS
     (RECORDS DELIMITED BY NEWLINE
       FIELDS (employee_number CHAR(2),
               employee_dob CHAR(20),
               employee_last_name CHAR(18),
               employee_first_name CHAR(11),
               employee_middle_name CHAR(11),
               employee_hire_date CHAR(10) date_format DATE mask "mm/dd/yyyy"
             )
```

```
        )
        LOCATION ('info.dat')
);
```

Prethodna naredba definiše eksternu tabelu `emp_load` uz pomoć koje se učitavaju podaci iz datoteke `info.dat` u tabelu `emp` u bazi. Podrazumeva se da je objekat direktorijuma `def_dir1` definisan u bazi i da je dostupan alat SQL Loader. Naredbom `LOCATION` ukazujemo nad kojom datotekom je definisana eksterna tabela.

4.6 Problemi prilikom migracije i njihovo rešavanje

U mnogim aspektima različiti relacioni sistemi su dosta slični, kao na primer: koncepti čuvanja podataka na fizičkom disku, tehnike upravljanja podacima, interfejsi za pristupanje podacima itd. S druge strane su primetne i značajne razlike koje se ogledaju u sledećim svojstvenim karakteristikama jednog RSUBP-a:

- Konvencije za davanje imena objektima
- Rukovanje NULL vrednostima
- Prepoznavanje veličina slova
- Pristup prilikom zaključavanja redova podataka
- Funkcionalnosti koje su podržane u programima koji rade nad bazom

Opisani proces migracije sa DB2 na Orakl bazu, primenom SQL Developer-a, omogućuje migraciju šeme DB2 baze pri čemu je moguće migrirati sledeće objekte: tabele, poglede, indekse, korisnike, ograničenja. Prilikom migracije ovih objekata potrebno je obratiti pažnju na sledeće probleme koji se javljaju kao posledica različitosti konvencija za davanje imena objektima u RSUBP-ovima Orakl i DB2:

- Korišćenje specijalnih karaktera
- Korišćenje rezervisanih reči u nazivima i definicijama objekata
- Osetljivost na veličinu slova u nazivima objekata
- Dozvoljena dužina naziva objekata

Prilikom migracije objekata šeme SQL Developer daje podršku za automatsko identifikovanje i rešavanje prethodno navedenih mogućih uzroka nepravilnosti koje se mogu pojaviti. Asistencija koju pruža SQL Developer u obavljanju ovog posla se ogleda u pravljenju izveštaja u kome je navedena lista objekata u čijim nazivima je se pojavila neka nepravilnost.

U Orakl bazi, u nazivima objekata kao prvo slovo ne može biti nijedan iz grupe specijalnih karaktera (#, %, $, _, itd). Neki od prethodno navedenih specijalnih karaktera (#, $, _) se mogu koristiti u nazivima objekata ali ne kao prvo slovo. S druge strane, u DB2 bazi je moguće koristiti specijalne karaktere kao prvo slovo, s tim što je zabranjena samo upotreba specijalnog karaktera _ (donja crta) i cifara od 0 do 9. Prilikom migracije SQL Developer rešava ove razlike na dva načina. Jedan od načina je zamena specijalnog karaktera nekim drugim znakom, dok je drugi način navođenje naziva objekata pod duplim navodnicima. Posle izvršenih prepravki nad objektima čiji nazivi nisu u skladu sa konvencijom za davanje imena Orakl baze, SQL Developer napravi izveštaj u kome su prikazani svi objekti koji su preimenovani prilikom migracije.

Ukoliko iz izvorne DB2 baze na Orakl bazu migriraju tabele koje u svom nazivu ili nazivu svojih kolona imaju neku rezervisanu reč čija upotreba nije dozvoljena u Orakl bazi onda se ovaj problem prevazilazi na sličan način kao i prethodni koji se odnosi na upotrebu specijalnih karaktera u nazivu objekata. Takve tabele, odnosno njihove kolone se ili preimenuju ili se njihov naziv stavlja pod navodnike.

Pojedini relacioni sistemi, u koje spada i DB2, dozvoljavaju upotrebu slova različite veličine u nazivima svojih objekata. Ukoliko se vrši migracija objekata na Orakl bazu koji u svojim nazivima imaju slova različite veličine onda u takvim slučajevima treba biti posebno pažljiv. Posle završetka procesa migracije u Orakl bazi su objekti sačuvani tako što su njihova imena zapisana velikim slovima, dok sa druge strane pretraga objekata se vrši na takav način da nije bitna veličina slova. Ukoliko je važno da objekti, sa mešovitim veličinama slova u svom nazivu, zadrže to svojstvo i u Orakl bazi onda oni moraju biti ručno napravljeni u skladu sa svojim definicijama u izvornoj DB2 bazi. Primer pravljenja jedne tabele, u Orakl bazi, koja u svom nazivu i u nazivu svojih kolona ima slova različite veličine:

```
create table "MojaTabela" ("Col1_id" number (10), "Col2" varchar2(50));
```

Kada se vrši pretraga objekata ovog tipa u Orakl bazi potrebno je eksplicitno navoditi njihove nazive i nazive njihovih kolona pod dvostrukim navodnicima. Na primer jedan SELECT upit bi bio:

```
select * from "MojaTabela" where "Col1"=1230;
```

Orakl baza ograničava dužinu imena objekata na maksimalno 30 karaktera. Drugi relacioni sistemi, u koje spada i DB2 dozvoljavaju da imena objekata imaju maksimalno 128 karaktera. Kao posledica ove restrikcije od strane Orakl baze pri migraciji objekti koji imaju u svom nazivu više od 30 karaktera se preimenuju. Ovakav pristup može imati uticaja na aplikacije koje se služe ovim objektima, tako da sve promene moraju biti ažurirane u njihovom kodu.

4.6.1 Identificirajuće kolone

DB2 relacioni sistem za upravljanje bazom podataka podržava koncept identificirajućih kolona u tabelama. Glavna prednost korišćenja kolona ovog tipa u tabelama ogleda se u pravljenju jedinstvenih vrednosti, od strane DB2 RSUBP-a, za ovu kolonu prilikom dodavanja novog reda podataka u tabelu. Ovakav pristup je prilično od koristi programerima koji zahvaljujući ovoj funkcionalnosti ne moraju da gube vreme za implementaciju dodatnog koda koji bi imao istu funkcionalnost.

Migracija tabele, iz izvorne DB2 baze, sa kolonom nad kojom je definisano ovo svojstvo nije podržana u Orakl RSUBP-u. Da bi se dobila funkcionalnost dodavanja jedinstvenih vrednosti za svaki novi red podataka u tabeli u Orakl bazi, potrebno je implementirati indirektni mehanizam koji koristi objekte sekvenci. Definicija jedne tabele u DB2 bazi koja ima definisanu identificirajuću kolonu bi bila:

```
create table tabela1
   (kol1 BIGINT Not Null Primary Key generated always as identity,
    Kol2 char(30) not null);
```

Prethodna funkcionalnost može biti implementirana u Orakl bazi na dva načina. Jedan pristup bi se sastojao iz korišćenja sekvence i definisanog okidača nad tabelom koji čine mehanizam za pravljenje

jedinstvenih vrednosti za svaki novi red koji se dodaje u tabelu. Prethodni mehanizam možemo implementirati na sledeći način:

Najpre definišemo tabelu sa definisanom numeričkom kolonom koja će prihvatati napravljene jedinstvene vrednosti za svaki novi red.

```
CREATE TABLE pomocna (
kol1 number(32) NOT NULL PRIMARY KEY, kol2 VARCHAR2(50) NOT NULL);
```

Sledeći korak je definisanje objekta sekvence koji će praviti novu jedinstvenu vrednost za svaki novi red koji se dodaje u tabelu.

```
CREATE SEQUENCE pom_sek START WITH 1 INCREMENT BY 1;
```

Potom se definiše okidač nad tabelom koji će popuniti identificirajuću kolonu sa jedinstvenim vrednostima koje napravi objekat sekvence.

```
CREATE OR REPLACE TRIGGER pom_okidac
BEFORE INSERT ON pomocna
FOR EACH ROW
WHEN (new.kol1 IS NULL)
BEGIN
       SELECT pom_sek.NEXTVAL
       INTO :new.kol1
       FROM dual;
END;
```

Implementacija ovog pristupa kao krajnji rezultat može dati povećan broj objekata Orakl baze kao što su sekvence i okidači, što može imati negativan uticaj na performanse prilikom učitavanja velike količine podataka u tabelu. Da bi se prevazišao ovaj problem, najčešća praksa prilikom migracije jedne tabele i njenih podataka jeste da se isključe sva ograničenja, okidači i indeksi definisani nad njom. Posle izvršene migracije tabele i njenih podataka, sekvenca se resetuje tako da njena početna vrednost odgovara maksimalnoj i uključi se odgovarajući okidač nad tabelom. Posle toga, za svaki novi red koji se dodaje u tabelu, zahvaljujući implementiranom mehanizmu, napravi se jedinstvena vrednost za idenificirajuću kolonu.

Drugi način implementacije identificirajuće kolone u Orakl bazi ogleda se u pristupu po kome objekti sekvenci naprave jedinstvene vrednosti, direktno iz INSERT naredbe prilikom dodavanja novog reda u tabelu. Ovaj pristup podrazumeva definisanje posebnog objekta sekvenci za svaku tabelu koja zahteva različit skup napravljenih jedinstvenih vrednosti. Umesto pravljenja okidača koji definišu jedinstvenu vrednost iz objekta sekvenci, moguće je direktno uključiti sekvencu kao što je prikazano u sledećoj SQL naredbi:

```
INSERT INTO pomocna (kol1, kol2) VALUES (pom_sek.NEXTVAL, 'nova vrednost');
```

Ovaj pristup zahteva dosta izmena ukoliko se često koriste identificirajuće kolone u tabelama DB2 baze koja migrira na Orakl RSUBP. Zbog toga je on preporučen ukoliko se identificirajuće kolone koriste u samo nekoliko tabela izvorne baze.

4.6.2 Rukovanje sa praznim stringovima i NULL vrednostima

U DB2 bazi postoji podrška za vrednost praznog stringa koju može imati jedna kolona karakter tipa i koja se razlikuje od NULL vrednosti. Nad ovom vrednošću se može primeniti operator poređenja na isti način kao i nad NULL vrednošću:

`Kolona1 = NULL` odnosno `Kolona1 = ''`

Dakle, kada je jednoj koloni karakter tipa u DB2 bazi pridružena vrednost praznog stringa to nije isto kao da joj je pridružena NULL vrednost.

U Orakl bazi ne postoji podrška za rad sa praznim stringovima. Oni se tretiraju kao NULL vrednosti. Ukoliko se u nekoj SELECT naredbi koristi poređenje `Kolona1 = ''` onda neće biti vraćen niti jedan red podataka.

Kako Orakl baza ne podržava rad sa praznim stringovima, pre migracije na izvornoj DB2 bazi se izvrši ažuriranje svih vrednosti praznih stringova sa NULL vrednostima.

4.6.3 Konverzija klasterovanih indeksa

Relacioni sistemi za upravljanje bazom podataka imaju svoje definicije klasterovanih indeksa kao objekata baze koje se mogu međusobno razlikovati u odnosu na cilj i namenu njihove implementacije.

U DB2 bazi, klasterovani indeks se pravi korišćenjem CLUSTER opcije i on obezbeđuje klasterovanje tabele nad kojom je definisan. Drugim rečima rečeno, sa klasterovanim indeksom definisanim nad tabelom, podaci u toj tabeli su organizovani na onaj način kako to definiše indeks. Jedna tabela može imati samo jedan definisan klasterovani indeks. Ovako definisani indeks nad tabelom obezbeđuje poboljšanje performansi tako što definiše na koji način jedan upit treba da pretražuje podatke. Pretraživanje podataka upit vrši u skladu sa njihovom organizacijom koja je definisana pomoću klasterovanog indeksa. Prilikom dodavanja novog reda u tabelu on se fizički čuva bilzu onih redova podataka čije ključne vrednosti su njemu bliske na osnovu njihove organizacije koju definiše klasterovani indeks.

U Orakl bazi klasterovani indeksi imaju drugačije značenje. Pod klasterovanim indeksom se smatra onaj indeks koji je definisan nad tabelom koja je klasterovana ili particionisana. Da bi se dobila DB2 funkcionalnost sa klasterovanim indeksima u Orakl bazi možemo iskoristiti indeksno organizovane tabele ili tabelarne klastere.

Indeksno organizovane tabele (IOT) se razlikuju od običnih tabela u odnosu na to kako su podaci u njima organizovani. Obične tabele u Orakl bazi su organizovane u vidu hip strukture. Prilikom dodavanja novih podataka u obične tabele, pristup je da se oni čuvaju u bilo kom bloku u kome postoji raspoloživ slobodan prostor pri čemu se popuna vrši proizvoljnim redosledom. S druge strane podaci u indeksno organizovanim tabelama se čuvaju u strukturi B drveta koje je logički organizovano u redosledu definisanom od strane indeksa nad primarnim ključem. Za razliku od običnih indeksa definisanih nad primarnim ključem, koji čuvaju samo kolone koje su uključene u njegovu definiciju, IOT indeksi čuvaju sve kolone tabele.

```
CREATE TABLE countries_demo
    ( country_id CHAR(2)
      CONSTRAINT country_id_nn_demo NOT NULL
      , country_name VARCHAR2(40)
      , currency_name VARCHAR2(25)
      , currency_symbol VARCHAR2(3)
      , region VARCHAR2(15)
      , CONSTRAINT country_c_id_pk_demo
            PRIMARY KEY (country_id ) )
      ORGANIZATION INDEX
      INCLUDING country_name
      PCTTHRESHOLD 2
      STORAGE
      ( INITIAL 4K )
      OVERFLOW
      STORAGE
           ( INITIAL 4K );
```

Tabelarni klasteri omogućavaju da blokovi podataka budu deljeni između tabela koje dele zajedničke kolone kako bi mogle biti grupisane zajedno. Ova obezbeđena funkcionalnost od strane Orakl baze nije ekvivalentna DB2 klasterovanim indeksima, ali poseduje dodatnu opciju koja omogućuje optimizaciju pretraživanja grupisanih tabela tako što čuva one podatke na osnovu kojih se spajaju tabele u istim blokovima. Klaster takođe može biti definisan nad jednom tabelom. Nije preporučeno da se tabelarni klasteri definišu nad tabelama u koje se često dodaju novi redovi podataka. Oni su više preporučeni za statičke tabele sa predefinisanim pristupom podacima u njima.

```
CREATE CLUSTER emp_dept (deptno NUMBER(3))
       SIZE 600
       TABLESPACE users
       STORAGE (INITIAL 200K
        NEXT 300K
        MINEXTENTS 2
        PCTINCREASE 33);

CREATE TABLE emp (
       empno NUMBER(5) PRIMARY KEY,
       ename VARCHAR2(15) NOT NULL,
       . . .
       deptno NUMBER(3) REFERENCES dept)
       CLUSTER emp_dept (deptno);

CREATE TABLE dept (
       deptno NUMBER(3) PRIMARY KEY, . . . )
       CLUSTER emp_dept (deptno);
```

5. Konverzija procedura, funkcija i okidača

Procedure, funkcije i okidači predstavljaju objekete koji se čuvaju u bazi i kao takvi pružaju posebne pogodnosti pri radu sa podacima iz baze. Ove pogodnosti se ogledaju u objektno orijentisanom pristupu, prilikom rada sa ovim objektima, koji se zasniva na konceptima enkapsulacije i mogućnosti ponovnog korišćenja jednom definisanog i sačuvanog objekta u bazi. Objektno orijentisani koncept enkapsulacije obezbeđuje da sva poslovna logika bude pokrivena kroz implementaciju ovih objekata i sačuvana na jednom mestu u bazi. Ovim pristupom se izbegava konfuzna situacija u kojoj bi poslovna logika bila pokrivena sa više različitih aplikacija napisanih u različitim programskim jezicima. Koncept ponovnog korišćenja jednom definisanog, napravljenog i na kraju sačuvanog objekta u bazi se zasniva na mogućnosti pristupa tom objektu od strane bilo koje aplikacije ili korisnika koji rade nad bazom.

Migracija ovih objekata je vremenski zahtevna i često komplikovana jer svaki proizvođač sistema za upravljanje bazom podataka obezbeđuje specifičan, proceduralno nadograđeni SQL, programski jezik u kome su procedure, funkcije i okidači napisani. Proizvođači RSUBP-ova ne obezbeđuju alate koji bi automatski izvršili konverziju definicija ovih objekata u njihov specifični, proceduralno nadograđeni SQL programski jezik, tako da se njihova konverzija zasniva na ručnim izmenama. Da bi se definicija jedne procedure, funkcije ili okidača prepisala iz jednog programskog jezika u drugi potrebno je da programer koji obavlja ovaj posao bude dobro upoznat sa sintaksom i semantikom oba jezika.

5.1 Pregled PL/SQL i SQL PL programskih jezika

Programski jezici SQL PL i PL/SQL predstavljaju proceduralnu nadogradnju SQL upitnog jezika. Programske jedinice napisane u ovim jezicima sastoje se od naredbi i drugih jezičkih elemenata koje se koriste kako bi se implementirala proceduralna logika nad SQL naredbama. Ovi jezici takođe obezbeđuju naredbe za deklarisanje promenljivih, kontrolu toka, dodeljivanje vrednosti promenljivim.

Za Orakl bazu je specifičan PL/SQL programski jezik, koji predstavlja Orakl specifičnu proceduralnu nadogradnju SQL programskog jezika. Sličan pristup, prilikom razvoja proceduralne nadogradnje na SQL programski jezik su imali i IBM-ovi inženjeri prilikom definisanja SQL PL programskog jezika. Između PL/SQL i SQL PL programskih jezika moguće je naći dosta ekvivalentnosti i preslikavanja. Ono što je takođe zajedničko za ova dva programska jezika jeste da je veoma lako razvijati u njima procedure i funkcije koje imaju dosta dobre performanse pri radu nad podacima u bazi.

Zbog razlike u pristupu prilikom implementacije procedura, funkcija i okidača u DB2 bazi, pojedini elementi SQL PL jezika se ne mogu koristiti u implementaciji funkcija i okidača. Dok se za implementaciju procedura u DB2 bazi koristi SQL PL programski jezik, za implementaciju funkcija i okidača se koristi njegov podskup takozvani “inline” SQL PL. Uz pomoć “inline” SQL PL-a se mogu implementirati i složene SQL naredbe koje se mogu izvršavati nezavisno ili se mogu iskoristiti za implementaciju tela funkcije ili okidača.

Ono što je karakteristično za programski jezik PL/SQL jeste njegova blokovska struktura. Osnovne programske jedinice napisane u ovom programskom jeziku predstavljaju zapravo blokove sa dodeljenim

nazivom ili bez njega. Blokovi bez naziva predstavljaju anonimne blokove dok oni sa nazivom predstavljaju procedure, funkcije ili okidače.

Anonimni blokovi predstavljaju osnovnu i najednostavniju formu PL/SQL koda. Omogućuju upotrebu proceduralne logike prilikom pravljenja koda bez njegovog čuvanja u vidu objekta u bazi. Ukoliko bismo tražili njima odgovarajuće SQL PL programske jedinice onda bi to bile složene SQL naredbe.

5.2 Konverzija procedura

Iako se procedure i funkcije na sistemima za upravljanje bazom podataka kao što su Orakl i DB2 mogu pisati u programskim jezicima Java, C, C++, Fortran, Cobol za pravljenje ovih objekata preporučeno je korišćenje proceduralno nadograđenog programskog jezika SQL koji je specifičan za oba RSUBP-a.

U daljem tekstu će biti opisano definisanje SQL PL procedura, sličnosti i razlike, pri obavljanju istog posla pravljenja procedura u Orakl PL/SQL programskom jeziku.

5.2.1 Sintaksa za definisanje procedure

Osnovna sintaksa pri definisanju SQL PL procedure jeste sledeća:

```
CREATE PROCEDURE naziv_procedure [( {parametri procedure} )]
[opcioni atributi procedure]
Begin
<naredbe>
End
```

U osnovi sintaksa za definisanje procedure u programskom jeziku PL/SQL je veoma slična. Razlike se ogledaju u postojanju OR REPLACE klauze koja se koristi pri prepisivanju postojeće procedure ukoliko je njena definicija samo izmenjena i klauze IS | AS koja služi da naznači početak sekcije za deklaraciju promenljivih. Takođe u telu PL/SQL procedure blok za upravljanje izuzecima mora biti naznačen sa ključnom rečju EXCEPTION.

```
CREATE [OR REPLACE] PROCEDURE naziv_procedure [( {parametri procedure} )]
[opcioni atributi procedure]
IS|AS
<sekcija za deklaraciju promenljivih>
BEGIN
<naredbe>
EXCEPTION
<blok za upravljanje izuzecima>
END
```

5.2.2 Tipovi parametara

Procedure napisane u SQL PL i PL/SQL programskim jezicima mogu imatu tri različita tipa parametara. Za definisanje kog tipa je parametar procedure koriste se sledeće ključne reči:

- IN – ulazni parametar
- OUT – izlazni parametar
- INOUT – označava da parametar može biti ulazni i izlazni

Upotreba prethodno navedenih ključnih reči pri definisanju SQL PL procedure bi bila sledeća:

```
CREATE PROCEDURE proc(IN p1 INT, OUT p2 INT, INOUT p3 INT)
```

Na sličan način se definiše tip parametra i u PL/SQL procedurama:

```
CREATE PROCEDURE OR REPLACE proc(p1 IN INTEGER, p2 OUT INTEGER, p3 INOUT
INTEGER)
```

Sintaksa za poziv prethodno definisane procedure je:

- za SQL PL proceduru: `CALL proc (10,?,4)`
- za PL/SQL proceduru:
```
VARIABLE p2 INTEGER
EXECUTE proc (10,:p2,4)
```

Znak ? u pozivu SQL PL procedure označava izlazni parametar. Što se tiče poziva PL/SQL procedure, najpre je potrebno deklarisati promenljivu a potom u pozivu procedure pomoću znaka : naznačiti da je u pitanju izlazna promenljiva.

5.2.3 Navođenje komentara u procedurama

Navođenje komentara u kodu procedure je istovetno bez obzira da li se radi o SQL PL ili PL/SQL proceduri. Komentari se mogu navoditi na sledeća dva načina:

- Tipični SQL komentar od jednog reda

  --Ovo je tipični SQL komentar

- Komentari od više redova se navode koristeći C stil pri navođenju komentara

  /* Ovo je tipični C komentar od više redova*/

5.2.4 Deklarisanje promenljivih

Deklarisanje promenljivih u SQL PL procedurama se vrši pomoću DECLARE naredbe. Promenljive se deklarišu u sklopu izvršnog bloka i sintaksa je sledeća:

```
DECLARE naziv_promenljive <tip_podatka> [DEFAULT vrednost];
```

Pristup deklarisanju promenljivih u PL/SQL procedurama je drugačiji. Promenljive se definišu u delu za deklaraciju promenljivih koji je naznačen sa IS|AS ključnom rečju:

```
CREATE [OR REPLACE] PROCEDURE naziv_procedure [( {parametri procedure} )]
[opcioni atributi procedure]
IS|AS
<sekcija za deklaraciju promenljivih>
naziv_promenljive <tip_podatka> [DEFAULT vrednost];
…
BEGIN
<naredbe>
EXCEPTION
<blok za upravljanje izuzecima>
END
```

Prilikom dodeljivanja vrednosti, prethodno deklarisanim promenljivim u SQL PL i PL/SQL procedurama mogu se takođe uočiti izvesne razlike. U SQL PL proceduri, prethodno definisanoj promenljivoj p1 se vrednost može dodeliti na jedan od sledećih načina:

```
SET p1 = 100;
```

```
VALUES(100) INTO p1;
```

Prethodno navedene naredbe su ekvivalentne i obe promenljivoj p1 dodeljuju vrednost 100. Pored ova dva načina promenljivoj p1 se može dodeliti i vrednost pomoću SELECT naredbe:

```
SET total = (select sum(c1) from T1);
```

Ovde treba biti siguran da upit vraća jedan red, inače će se javiti greška prilikom izvršavanja procedure.

U PL/SQL procedurama promenljive mogu biti deklarisane u sekciji za deklaraciju ili kasnije u izvršnom bloku. Prilikom deklarisanja promenljive koristi se naredba dodele :=. Primer dodeljivanja vrednosti 100 promenljivoj p1:

```
p1 := 100;
```

Promenljivoj se vrednost može dodeliti korišćenjem SELECT INTO naredbe na sledeći način:

```
SELECT SUM(c1) INTO p1 FROM t1;
```

5.2.5 Rad sa kursorima

Kursor predstavlja mehanizam uz pomoć koga se rukuje sa rezultujućim skupom koji je vraćen izvršavanjem jednog upita. U SQL PL procedurama kursor se definiše koristeći sledeću sintaksu:

```
DECLARE <naziv_kursora> CURSOR [WITH RETURN <return target>]
<SELECT naredba>;
```

Prilikom definisanje kursora u SQL PL procedurama može se koristiti opciona klauza WITH RETURN koja služi da naznači kome se vraća rezultujući skup kojim rukuje definisani kursor. Vrednosti koje se mogu koristiti u ovoj klauzi su:

- CLIENT – rezultujući skup će se vratiti klijentskoj aplikaciji iz koje je izvršen poziv
- CALLER – rezultujući skup će će se vratiti pozivaocu procedure

Pored naredbe za definisanje kursora u SQL PL programskom jeziku se koriste i naredbe za rad sa kursorima kao što su naredba za otvaranje kursora, za uzimanje vrednosti iz kursora i naradba za zatvaranje kursora.

```
OPEN <naziv_kursora>;
FETCH <naziv_kursora> INTO <promenljiva>;
CLOSE <naziv_kursora>;
```

Kursori u PL/SQL procedurama se deklarišu u sekciji za deklaraciju koristeći veoma sličnu sintaksu prethodno navedenoj:

```
CURSOR naziv_kursora [(parametri kursora)]
IS select naredba;
```

Takođe su podržane i naredbe za rad sa kursorima koje su identične prethodno navedenim naredbama koje se koriste u SQL PL proceurama.

Kursori se posebno pokazuju korisnim u procedurama koje kao povratnu vrednost vraćaju više redova podataka. Sledeća dva primera ukazuju na razlike koje se mogu uočiti pilikom upotrebe kursora u SQL PL i PL/SQL procedurama koje kao povratnu vrednost vraćaju više redova podataka.

Primer korišćenja kursora u SQL PL proceduri koja vraća više redova:

```
CREATE PROCEDURE DEC_RTN_RECORDSET
(
IN  P_InvoiceDate   TIMESTAMP
)
DYNAMIC RESULT SETS 1
LANGUAGE SQL

BEGIN
DECLARE c2 CURSOR WITH RETURN TO CLIENT FOR
SELECT Invoice_Number, Invoice_Date, Client_ID, Invoice_Amt
FROM Invoice
WHERE Invoice_date <= p_InvoiceDate
ORDER BY Invoice_number;

OPEN c2;
END;
```

Kursor se u SQL PL procedurama najpre deklariše, pri čemu se definiše kome će rezultujući skup na koji ukazuje kursor biti vraćen, da li pozivaocu procedure ili klijentskoj aplikaciji koja poziva proceduru. Na kraju se otvara kursor naredbom OPEN.

Prilikom korišćenja kursora u PL/SQL proceduri koja vraća više redova najpre je potrebno definisati referentnu kursor promenljivu:

```
CREATE OR REPLACE PACKAGE types
AS
type cursorType is ref cursor;
END
```

Za Orakl bazu je specifična kursor promenljiva koja je tipa REF CURSOR i koja se korist prilikom prosleđivanja rezultujućeg skupa podataka između rutina u bazi ili različitih klijentskih aplikacija. Kao i kursori i kursor promenljive ukazuju na trenutni red u rezultujućem skupu podataka jednog upita. Razlika između kursora i kursor promenljivih je ista kao i razlika između konstanti i promenljivih. Kursori su statički za razliku od kursor promenljivih koje su dinamičke jer nisu vezane za specifični upit.

Primer jedne PL/SQL procedure u kojoj se koristi kursor promenljiva kao izlazni parametar:

```
CREATE OR REPLACE PROCEDURE DEC_RTN_RECORDSET
(
p_InvoiceDate      IN   DATE,
```

```
p_ResultSet        OUT  TYPES.cursorType
)
AS
BEGIN
OPEN p_ResultSet FOR
SELECT Invoice_Number, Invoice_Date, Client_ID, Invoice_Amt
FROM Invoice
WHERE Invoice_date <= p_InvoiceDate
ORDER BY Invoice_number;

END DEC_RTN_RECORDSET;
```

5.2.6 Kontrola toka i uslovne naredbe

Kao i u svim drugim i u SQL PL I PL/SQL programskim jezicima se mogu koristiti naredbe za kontrolu toka i uslovne naredbe kao što su: FOR, WHILE, LOOP, CASE, IF, RETURN itd.

Prilikom poziva ugnježdene procedure u SQL PL procedurama je potrebno koristiti CALL naredbu. Ovo nije slučaj sa PL/SQL procedurama, gde se ugnježdena procedura poziva standardno, navođenjem njenog imena i odgovarajućih parametara.

5.2.7 Rukovanje izuzecima

U oba jezika, u PL/SQL-u i SQL PL-u postoji podrška za rukovanje izuzecima koji se mogu javiti tokom izvršavanja procedura, funkcija, okidača. Mehanizam rukovanja izuzecima u ovim objektima nastoji da razdvoji obrade grešaka od glavne logike koja je pokrivena njihovom implementacijom.

U programskom jeziku PL/SQL u okviru izvršnog bloka procedura, funkcija, okidača postoji posebna sekcija za rukovanje izuzecima i njena sintaksa je:

```
EXCEPTION
   WHEN nazivi_izuzetka1 THEN <izvršne naredbe>
   WHEN nazivi_izuzetka2 THEN <izvršne naredbe>
   ...
   WHEN nazivi_izuzetkaN THEN <izvršne naredbe>
   WHEN OTHER  <izvršne naredbe> ;
```

Gde naziv izuzetka može biti jedan od predefinisanih izuzetaka kao što su NO_DATA_FOUND, TOO_MANY_ROWS ili ti izuzeci mogu biti korisnički definisani sledećom naredbom:

```
naziv_izuzetka EXCEPTION;
   PRAGMA EXCEPTION_INIT(naziv_izuzetka, SQLCODE);
```

U DB2 bazi, rukovanje izuzecima u procedurama je postignuto kroz korišćenje mehanizma za rukovanje uslovima. Gde mehanizam za rukovanje uslovima predstavlja SQL naredba koja se izvršava kada je specijalni uslov ispunjen za vreme izvršavanja naredbi u telu procedure.

Za deklaraciju mehanizma za rukovanje uslovima se koristi sledeća sintaksa:

```
DECLARE {CONTINUE | EXIT | UNDO} HANDLER FOR <uslov>
            SQL-procedure-naredba;
```

Gde uslov može biti jedan od sledećih:

- SQLSTATE vrednost
- SQLEXEPTION (SQLCODE < 0)
- SQLWARNING (SQLCODE > 0)
- NOT FOUND
- Naziv uslova

U PL/SQL rutinama u okviru izvršne sekcije postoji poseban blok za rukovanje izuzecima. Početak ovog bloka je naznačen sa ključnom rečju EXCEPTION.

```
EXCEPTION
  WHEN NO_DATA_FOUND THEN v_status         :=0;
  WHEN OTHER             THEN v_err_flag :=1;
```

U SQL PL kodu procedure da bi se implementirao prethodni mehanizam za rukovanje izuzecima potrebno je definisati dve sledeće naredbe:

```
DECLARE CONTINUE HANDLER FOR NOT FOUND
  SET v_status   = 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
  SET v_err_flag = 1;
```

U prethodnom primeru naziv Orakl izuzetka odgovara DB2 nazivu uslova i predefinisani Orakl NO_DATA_FOUND izuzetak odgovara DB2 NOT FOUND uslovu. Da bi se međusobno preslikali drugi Orakl predefinisani izuzeci, odgovarajuće DB2 SQLSTATE vrednosti moraju biti iskorišćene pri čemu se uslov definiše koristeći sledeću sintaksu:

```
DECLARE  condition_name CONDITION FOR SQLSTATE value;
```

Na primer, Orakl predefinisani izuzetak TOO_MANY_ROWS može biti konvertovan koristeći sledeću naredbu:

```
DECLARE too_many_rows CONDITION FOR SQLSTATE '21000';
```

Odgovarajući kod(HANDLER) koji rukuje ovim uslovom može biti definisan na sledeći način:

```
DECLARE EXIT HANDLER FOR too_many_rows
BEGIN
...
END;
```

Za razliku od Orakl RSUBP-a, rukovanje izuzecima u funkcijama i okidačima nije podržano u DB2 bazi. Konverzija Orakl funkcije koja sadrži sekciju za rukovanje izuzecima, se vrši tako što se ona konvertuje u proceduru sa obaveznim jednim izlaznim parametrom u DB2 bazi.

5.2.8 Dinamički SQL

Pored statičkih SQL naredbi u SQL PL i PL/SQL procedurama se može koristiti i dinamički SQL čija struktura naredbe nije poznata sve do vremena izvršavanja programa. Na primer, ukoliko su col1 i naziv_tabele promenljive onda definišemo dinamičku SQL naredbu na sledeći način:

```
'SELECT ' || col1 || ' FROM ' || tabname;
```

Pristup prilikom izvršavanja dinamičkih SQL naredbi je gotovo istovetan u SQL PL i PL/SQL procedurama. U oba slučaja izvršavanje dinamičkih SQL naredbi se može izvršiti korišćenjem EXECUTE IMMEDIATE naredbe.

Primer korišćenje EXECUTE IMMEDIATE naredbe iz izvršne sekcije SQL PL procedure prilikom izvršavanja dinamičke SQL naredbe:

```
BEGIN

DECLARE stmt varchar(255);
DECLARE col1 varchar(255)= ‘col1’;
DECLARE tabname varchar(255)= ‘tabname’;

SET col1 = ‘col1’;
SET tabname = ‘tabname’;

SET stmt = ’ 'SELECT ' || col1 || ' FROM ' || tabname;’;

EXECUTE IMMEDIATE stmt;

END
```

Primer korišćenje EXECUTE IMMEDIATE naredbe iz izvršne sekcije PL/SQL procedure prilikom izvršavanja dinamićke SQL naredbe:

```
DECLARE

stmt varchar(255);
col1 varchar(255)= ‘col1’;
tabname varchar(255)= ‘tabname’;

BEGIN

col1 = ‘col1’;
tabname = ‘tabname’;

stmt = ‘'SELECT ' || col1 || ' FROM ' || tabname;’;

EXECUTE IMMEDIATE stmt;

END
```

Dinamički SQL se u SQL PL procedurama može takođe izvršiti korišćenjem PREPARE naredbe. Korišćenje ove naredbe je pogodno ukoliko se izvršava više SQL dinamičkih naredbi jedna za drugom. Upotreba PREPARE naredbe nije uobičajena u PL/SQL procedurama. Primer korišćenje PREPARE naredbe iz izvršne sekcije SQL PL procedure prilikom izvršavanja dinamičke SQL naredbe:

```
BEGIN

DECLARE stmt varchar(255);
DECLARE st STATEMENT;

SET stmt = 'INSERT INTO T2 VALUES (?, ?)';
PREPARE st FROM stmt;
```

```
EXECUTE st USING value1, value1;
EXECUTE st USING value2, value2;

END
```

5.3 Konverzija funkcija

U sistemima za upravljanje bazom podataka funkcije enkapsuliraju određenu poslovnu logiku i kao takve čuvaju se u bazi. Postoje različite primene funkcija pri radu sa podacima iz baze ili pri korišćenju funkcija od strane aplikacija koje rade nad bazom. Funkcije se mogu koristiti kao operatori nad podacima iz kolone tabele, kao proširena podrška za ugrađene funkcije, za poboljšanje sigurnosti baze itd.

U DB2 sistemu za upravljanje bazom podataka funkcije se mogu pisati u programskim jezicima SQL PL, PL/SQL, C/C++, Java. U okviru ovog dela će biti predstavljene SQL PL funkcije koje možemo, u zavisnosti od toga kakvu vrednost vraćaju, podeliti na skalarne i tabelarne funkcije. Ukoliko SQL PL funkcija vraća samo jedan red ili kolonu podataka nju takođe možemo definisati kao tabelarnu. Za DB2 funkcije I okidače karakteristično je da su napisane pomoću "inline" SQL PL – a koji predstavlja podskup SQL PL programskog jezika. Takođe će biti predstavljene sve sličnosti SQL PL funkcija sa PL/SQL funkcijama.

Korisnički definisane SQL PL skalarne funkcije vraćaju samo jednu vrednost i u njima je obavezno korišćenje RETURN klauze. Ove funkcije u svom telu ne mogu imati SQL naredbe koje menjaju stanje podataka u bazi, kao što su INSERT, UPDATE, DELETE. Pored korisninički definisanih u DB2 bazi moguće je koristiti i ugrađene skalarne funkcije kao što su SUM(), AVG(), DIGITS(), COALESCE(), SUBSTR() itd.

Prilikom poziva skalarne funkcije u DB2 bazi se koristi sledeća sintaksa:

```
db2 "values (naziv_funkcije(parametar))";
```

ili

```
db2 "select (naziv_funkcije (parametar)) from sysibm.sysdummy1";
```

Tabelarne funkcije u DB2 bazi su svojstvene po tome što kao rezultat mogu vratiti više redova, jedan red ili kolonu podataka. Za razliku od skalarnih funkcija u tabelarnim funkcijama je moguće koristiti SQL naredbe koje menjaju stanje podataka iz baze kao što su INSERT, UPDATE, DELETE. Za tabelarne funkcije možemo reći da su slične pogledima, međutim njihova je prednost i snaga, u odnosu na poglede, u tome što se u njima mogu koristiti DML naredbe. Takođe postoje i ugrađene tabelarne funkcije.

Primer jedne tabelarne SQL PL funkcije:

```
CREATE FUNCTION FN_GET_ROWS()
RETURNS TABLE
(
TEST_ID INT,
TEST_DATE TIMESTAMP
)
LANGUAGE SQL
READS SQL DATA
CALLED ON NULL INPUT
NO EXTERNAL ACTION
```

```
BEGIN ATOMIC
RETURN
SELECT TEST_ID, TEST_DATE
FROM TEST;
END@
```

Da bi se pozvala prethodna funkcija koristimo sledeću SELECT naredbu:

```
SELECT * FROM TABLE(FN_GET_ROWS()) AS T
```

Kao što je prikazano iznad tabelarne funkcije se pozivaju iz FROM klauze SELECT upita pomoću TABLE() ugrađene funkcije.

Što se tiče pristupa pri radu sa Orakl PL/SQL funkcijama on je nešto drugačiji u odnosu na pristup u radu sa funkcijama u DB2 bazi. Razlike se prevashodno ogledaju u tome što se u Orakl bazi mogu pisati funkcije u čijem se telu mogu koristiti sve naredbe PL/SQL jezika, dok se za definisanje tela funkcije u DB2 bazi koristi takozvani "inline" SQL PL koji predstavlja podskup SQL PL jezika.

PL/SQL funkcije predstavljaju imenovane blokove koji obavezno vraćaju vrednost i samim tim u njima mora biti RETURN klauza. Ovde definišemo funkcije kao imenovane PL/SQL blokove koji prihvataju parametre, mogu biti pozvani i na kraju vraćaju neku vrednost. One se najčešće pozivaju kao deo nekog izraza ili ukoliko treba da obezbede parametar nekom drugom podprogramu. Ovako definisane Orakl PL/SQL funkcije odgovaraju skalarnim SQL PL funkcijama.

Što se tiče DB2 koncepta tabelarnih funkcija, on se implementira na drugačiji način u Orakl bazi. Naime, implementacija funkcija koje vraćaju više redova se u Orakl PL/SQL programskom jeziku vrši pomoću tabelarnog objekat tipa.

Najpre se definiš objektni tip:

```
CREATE OR REPLACE TYPE TEST_OBJ_TYPE IS OBJECT
(
TEST_ID   NUMBER(9),
TEST_DESC VARCHAR(30)
);
```

Kada je objekat definisan, potom definišemo njegov tabelarni tip:

```
CREATE OR REPLACE TYPE TEST_TABTYPE AS TABLE OF TEST_OBJ_TYPE;
```

Potom definišemo funkciju koja vraća kao rezultat skup redova:

```
CREATE OR REPLACE FUNCTION FN_GET_ROWS
RETURN TEST_TABTYPE
AS
V_Test_Tabtype Test_TabType;
BEGIN
SELECT TEST_OBJ_TYPE(A.Test_Id, A.Test_Desc)
BULK COLLECT INTO V_Test_TabType
FROM
(SELECT Test_Id, Test_Desc
FROM Test
```

```
) A;
RETURN V_Test_TabType;

END;
```

Za razliku od Orakl baze u kojoj je dozvoljeno pisanje više naredbi u telu tabelarne funkcije, u DB2 bazi nije podržano definisanje tela tabelarne funkcije uz pomoć više naredbi, što svakako predstavlja veliko ograničenje prilikom definisanja funkcija ovog tipa.

5.4 Konverzija okidača

Okidači predstavljaju objekte baze koji se automatski izvršavaju kada je određeni uslov ispunjen. Oni se definišu nad određenom tabelom u bazi i okidaju se automatski kada se vrše okidajuće SQL naredbe, kao što su INSERT, UPDATE, DELETE nad podacima u tabeli.

U Orakl i DB2 bazi je podržana funkcionalnost okidača. U obe baze se okidači mogu koristiti za širok spektar aktivnosti, kao što su ažuriranja nad drugim tabelama, automatsko pravljenje ili transformisanje vrednosti učitanih ili ažuriranih redova u tabeli, pozivanje funkcija radi izvršavanja određenih poslova itd.

U Orakl i DB2 bazi okidači se mogu izvršavati po jednom za svaki red koji je pogođen okidajućom naredbom ili jednom u odnosu na određenu okidajuću naredbu. U zavisnosti od toga kada se izvršavaju razlikujemo tri tipa okidača:

- Okidač koji se izvršava pre izvršenja naredbe – Ovaj tip okidača se izvršava pre nego što su izvršene INSERT, UPDATE, DELETE okidajuće naredbe.
- Okidač koji se izvršava posle izvršenja naredbe - Ovaj tip okidača se izvršava pošto su izvršene INSERT, UPDATE, DELETE okidajuće naredbe.
- “Instead of” – Ovaj tip okidača se definiše nad pogledima. Kako su pogledi definisani koristeći neku SELECT naredbu nad jednom ili više tabela, podaci u pogledim ne mogu biti modifikovani. Koristeći ovaj tip okidača, kod korisnika koji modifikuje pogled se može stvoriti iluzija da je uspeo da izvrši modifikaciju podataka u njemu pomoću neke DML naredbe. Ovo se javlja kao posledica izvršavanja logike okidača umesto okidajuće naredbe.

Primer jednog DB2 okidača koji upisuje određene vrednosti iz izbrisanog reda jedne tabele u drugu istorijsku tabelu:

```
CREATE TRIGGER emp_history_trg
   AFTER DELETE ON EMPLOYEES
   REFERENCING OLD AS d
   FOR EACH ROW
BEGIN ATOMIC
   INSERT INTO emp_history ( emp_id
                            ,first_name
                            ,last_name)
   VALUES ( d.emp_id
           ,d.first_name
           ,d.last_name);
END;
```

Promenljiva definisana pomoću naredbe REFERENCING OLD AS predstavlja novu vrednost koja treba da bude učitana INSERT naredbom. Za slučaj da je potrebna nova vrednost prilikom učitavanja, do nje bismo mogli doći definisanjem promenljive pomoću `REFERENCING NEW AS naziv_promenljive` naredbe.

Odgovarajući okidač bi se definisao u Orakl bazi na sledeći način:

```
CREATE TRIGGER emp_history_trg
   AFTER DELETE ON employees
   FOR EACH ROW
BEGIN
   INSERT INTO emp_history( emp_id
                           ,first_name
                           ,last_name)
   VALUES ( :old.emp_id
            ,:old.first_name
            ,:old.last_name );
END;
```

Iz prethodnog primera se može primetiti da se u Orakl bazi koriste predefinisane OLD i NEW promenljive preko kojih se dolazi do starih odnosno novih vrednosti prilikom izvršavanja neke okidajuće DML naredbe.

6. Migracija JDBC aplikacija između RSUBP Orakl i DB2

JDBC (***Java Database Connectivity)*** API obezbeđuje standardni interfejs za pristup i rad sa podacima u bazi. Java kod koji se koristi klasama i interfejsima JDBC API-ja je lako prenosiv između različitih platformi odnosno RSUBP-ova nad kojima radi. Jedine izmene, koje je potrebni izvršiti, odnose se na JDBC drajver koji je potrebno učitati i na nisku za konekciju.

JDBC API omogućava Java programima da uspostave sesiju , izvršavaju SQL naredbe, preuzimaju rezultate izvršenih upita nad nekom relacionom bazom, nezavisno od toga od kog proizvođača je RSUBP koji upravlja bazom. U skladu sa time, JDBC API se sastoji od četiri glavne komponenete: JDBC drajvera, komponente za definisanje konekcije ka bazi, komponente za definisanje SQL naredbi i komponente za vraćanje rezultata upita izvršenih iz Java programa nad bazom. Proizvođači RSUBP-ova obezbeđuju drajvere koji su u skladu sa JDBC specifikacijom. Komponente koje se odnose na rad sa konekcijom, SQL naredbama, rezultujućim skupom izvršenih SQL naredbi su sastavni deo JDBC API paketa *java.sql*.

JDBC API obezbeđuje klase koje pružaju interfejs za rad sa prethodno navedenim komponenetama:

- `java.sql.Driver` i `java.sql.DriverManager` za upravljanje JDBC drajverima
- `java.sql.Connection` za korišćenje konekcije
- `java.sql.Statement` za definisanje i izvršavanje SQL upita
- `java.sql.ResultSet` za obradu rezultata izvršenog upita

Prilikom pisanja JDBC aplikacije, jedina potrebna specifična informacija koju zahteva drajver jeste URL baze. Moguće je napisati JDBC aplikaciju koja prihvata URL infomaciju tokom svog izvršavanja. Koristeći informacije koje se odnose na URL baze, naziv korisnika, lozinku korisnika JDBC aplikacija zahteva konekciju preko klase `java.sql.Connection`.

Tipični JDBC program se sastoji iz sledećeg niza koraka:

1. Učitavanje drajvera baze na koji se vrši konekcija
2. Uspostavljanje konekcije, koristeći JDBC URL za konekciju
3. Pravljenje i izvršavanje SQL naredbi
4. Korišćenje rezultujućeg skupa podataka vraćenih upitom
5. Zatvaranje konekcije

Prilikom konverzije JDBC programa sa jednog RSUBP-a na drugi posebno treba obratiti pažnju, zbog potrebnih izmena, na one delove koda koji se odnose na:

- JDBC drajver
- Niska za uspostavljanje konekcije
- Nekompatibilne SQL naredbe

U daljem tekstu će biti opisane razlike u kodu, prilikom učitavanja drajvera, jedne JDBC aplikacije koja migrira između Orakl i DB2 baze, razlike prilikom definisanja stringa za konekciju u odnosu na RSUBP i tip drajvera koji aplikacija koristi. Na kraju će biti opisane sve razlike koje se javljaju prilikom poziva procedura iz koda JDBC aplikacije.

6.1 Učitavanje JDBC drajvera

Koristeći odgovarajuće drajvere, Java program može da uspostavi konekciju ka relacionim bazama kojima upravljaju različiti RSUBP-ovi. Da bi jedan Java program upravljao operacijama sa drajverom, JDBC API obezbeđuje klasu za upravljanje drajverom, `java.sql.DriverManager` koja učitava drajver i uspostavlja konekciju ka bazi.

`DriverManager` klasa registruje drajver koji je potrebno koristiti u zavisnosi na kom RSUBP-u radi baza. Postoji nekoliko načina da se registruje drajver:

- Registrovanje drajvera eksplicitno:

  `DriverManager.registerDriver(`*`instanca drajvera`*`)`

  Gde instanca drajvera predstavlja instancu klase drajvera.

- Učitavanje klase drajvera koristeći:

  `Class.forName(`*`klasa drajvera`*`)`

  Gde klasa drajvera predstavlja JDBC klasu drajvera. Kada je učitan, svaki drajver se mora implicitno registrovati koristeći `DriverManager.registerDriver` method.

Na primer da bi se registrovao DB2 JDBC Type 4 drajver može se iskoristiti jedno od sledećeg:

```
DriverManager.registerDriver(new com.ibm.db2.jcc.DB2Driver);
```

ili

```
Class.forName("com.ibm.db2.jcc.DB2Driver");
```

Dok se za Orakl bazu može iskoristiti:

```
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver ());
```

ili

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

JDBC API od verzije 2.0 poseduje implementiran `DataSource` interfejs koji obezbeđuje visok nivo portabilnosti JDBC aplikacije nezavisno od RSUBP-a koji upravlja bazom podataka. Korišćenje klase `DriverManager` u aplikaciji smanjuje njenu portabilnost jer postoji potreba za identifikovanjem specifične klase JDBC drajvera i njegovog URL-a u zavisnosti od toga nad kojom bazom aplikacija radi. Upotreba `DataSource objekta` zamenjuje upotrebu `DriverManager` interfejsa.

Najednostavniji način upotrebe `DataSource` objekta jeste kada se on definiše i koristi u okviru aplikacije, na sličan način kao i `DriverManager` interfejs.

Za DB2 bazu DataSource objekat se definiše na sledeći način:

```
DB2SimpleDataSource dbds=new DB2SimpleDataSource();
```

Za Orakl bazu DataSource objekat se definiše na sledeći naćin:

```
OracleDataSource ods = new OracleDataSource();
```

Da bi se postigao najveći nivo portabilnosti potrebno je definisati a potom i upravljati `DataSource` objektom zasebno, nezavisno od aplikacije, pomoću aplikativnog servera. Program koji definiše i upravlja `DataSource` objektom koristi Java Naming and Directory Interface (JNDI) interfejs koji pridružuje logički naziv `DataSource` objektu. JDBC aplikacija, u ovom slučaju, koristi `DataSource` objekat na osnovu tog logičkog imena i ne zahteva nikakvu dodatnu informaciju o izvoru podataka. Ukoliko se u toku rada aplikacije izmene atributi izvora podataka, nije potrebno vršiti nikakve izmene u kodu aplikacije čime se postiže najveći nivo portabilnosti.

6.2 Definisanje niske za konekciju

Posle uočenih razlika pri registrovanju drajvera i rešavanja nekompatibilnosti, potrebno je obratiti pažnju na nisku za konekciju koji se prosleđuje drajveru. Ova niska se svakako razlikuje između RSUBP Orakl i DB2. Takođe postoje i različiti stringovi za jedan RSUBP u zavisnosti od toga koji se drajver koristi za uspostavljanje konekcije.

Ukoliko se za registrovanje drajvera koristi `DriverManager` interfejs onda se konekcija uspostavlja uz pomoć metode `getConnection`.

- Za DB2 bazu, ukoliko se koristi drajver tipa 4, string se definiše na sledeći način:

```
Connection conn =

 DriverManager.getConnection("jdbc:db2://naziv_baze","korisnik","šifra");
```

- Za Orakl bazu, ukoliko se koristi *thin* drajver, string se definiše na sledeći način:

```
Connection conn =

  DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl",
  "korisnik", "šifra");
```

Ukoliko se za registrovanje drajvera koristi `DataSource` objekat onda se konekcija uspostavlja tako što se ovom objektu prosleđuju parametri koje on prihvata svojim odgovarajućim metodama:

- Za DB2 bazu posle definisanja `DataSource` objekta parametri za uspostavljanje konekcije se prosleđuju na sledeći način:

```
db2ds.setDatabaseName("db2loc1");                          3

db2ds.setUser("korisnik");

db2ds.setPassword("šifra");
```

```
Connection con=db2ds.getConnection();
```

- Za Orakl bazu posle definisanja `DataSource` objekta parametri za uspostavljanje konekcije se prosleđuju na sledeći način:

```
String url = "jdbc:oracle:thin:korisnik/šifra@//localhost:1521/orcl";

ods.setURL(url);

conn = ods.getConnection();
```

6.3 Pozivi procedura i funkcija u JDBC programu

Nema velikih razlika u pristupu prilikom poziva procedura ili funkcija u JDBC programima koji rade nad Orakl i DB2 bazama. Nekompatibilnosti koje se javljaju su najčešće uzrokovane razlikama u SQL naredbama u telu procedure ili funkcije i preslikavanjima između Java tipova i tipova baze nad kojom aplikacija radi. Posle migracije baze Java tipove je potrebno preslikati u tipove koje podržava novi RSUBP koji upravlja migriranom bazom. Rukovanje sa ulaznim i izlaznim parametrima prilikom poziva procedure ili funkcije iz JDBC programa se takođe u određenim slučajevima mogu razlikovati.

Posmatrajmo slučaj kada se u JDBC programu koji radi nad Orakl bazom poziva funkcija koja kao povratnu vrednost vraća kursor.

```
CREATE TYPE CursorType IS REF CURSOR;

CREATE OR REPLACE FUNCTION funkcija(v_num IN INTEGER)
               RETURN CursorType
```

Nema razlike u pristupu pri pozivu procedure ili funkcije iz JDBC programa nad Orakl i DB2 bazama. Najpre se definiše string promenljiva pomoću specijalne sintakse, potom se definišu parametri pre nego što se procedura ili funkcija izvrši.

```
String SP_CALL = "{? := call funkcija(?)}";

Connection conn =
  DriverManager.getConnection (url, korisnik, šifra);

try {
    CallableStatement stmt = conn.prepareCall(SP_CALL);
    stmt.registerOutParameter (1, OracleTypes.CURSOR);
    stmt.setInt(2, 6);
    stmt.execute();
    ResultSet rs = (ResultSet) stmt.getObject(1);
    while(rs.next())
      {   // ... }
    }
```

Iz prethodnog koda možemo primetiti da se najpre inicijalizuje povratna vrenost Orakl funkcije, potom i ulazni parametar. Prilikom migracije JDBC aplikacije, na DB2 bazu, koja u svom kodu ima poziv prethodno opisane funkcije preporučeno je da se Orakl funkcija konvertuje u DB2 proceduru. Potreba za

ovom konverzijom se javlja jer DB2 JDBC drajver ima drugačiji implementiran pristup pri rukovanju sa ovakvim pozivima funkcija u JDBC programu.

Funkcija se može konvertovati u proceduru primenom tehnike “ogrtač funkcije”.

```
CREATE OR REPLACE PROCEDURE mojafunkcija_ogrtač(C OUT sys_refcursor ,arg1,
...argn ) IS
   BEGIN
      c:= mojafunkcija(arg1, ...argn  );
   END;
/
CREATE OR REPLACE FUNCTION mojafunkcija(arg1, ...argn) IS
   BEGIN
      ...
   ENd;
```

7. Migracija Orakl baze na DB2 RSUBP

Poslovi migracije baze podataka sa jednog RSUBP-a na drugi se tradicionalno dele u tri faze: migracija šeme, migracija podataka, migracija aplikacija. Za poslova konverzije šeme i migracije podataka dostupni su alati čijom upotrebom se postiže izvestan nivo automatizacije. Konverzija aplikacija predstavlja posao za koji je potrebno najviše truda i vremena. Nivo kompatibilnost DB2 RSUBP-a u odnosu na Orakl bazu predstavlja novi pristup u rešavaju problema migracije aplikacija. On se ogleda u njegovoj mogućnosti da podrži sva proširenja SQL jezika, tipove i strukture podataka koji su svojstveni za Orakl bazu. Zahvaljujući ovoj mogućnosti, postojeće aplikacije mogu, bez nekih značajnijih izmena u njihovom kodu, pristupati migriranoj Orakl bazi koja sada radi na novoj platformi. Proces migracije aplikacija nad bazom koja radi u novom okruženju je samim tim znatno brži i jednostavniji.

7.1 Kompatibilnost DB2 RSUBP- a za Orakl bazu

Od verzije 9.7 DB2 RSUBP obezbeđuje, preko ugrađenog nivoa kompatibilnosti, proširenu podršku za Orakl ekstenzije SQL programskog jezika, PL/SQL proceduralni programski jezik, Orakl tipove podataka, skalarne funkcije, konkurentnost, ugrađeni paketi, SQL *Plus interfejs. Ovakve karakteristike koje pruža RSUBP DB2 u velikoj meri doprinose mogućnosti razvoja aplikacija koje mogu da rade i na DB2 i na Orakl sistemima za upravljanje bazom podataka. Dodatno, IBM nudi dodatni skup alata za proces uključivanja Orakl aplikacija na DB2 sistemu. Njega čine visoko skalabilni IBM Data Movement alat za migraciju šeme i podataka baze kao i Optim Data Studio koji nudi mogućnost editovanja i profilisanja PL/SQL koda.

7.1.2 DB2 vektor kompatibilnosti

DB2_COMPATIBILITY_VECTOR je registarska promenljiva koja se koristi za uključivanje jedne ili više karakteristika kompatibilnosti DB2 sistema za Orakl bazu. U ovoj promenljivoj se čuva 12-bitna vrednost gde svaki bit odgovara jednoj vrsti DB2 kompatibilnosti. Da bi se dobila potpuna kompatibilnost DB2 sistema sa Orakl bazom potrebno je komandom *db2set* podesiti vrednost promenljive na *ORA***.** Takođe je moguće selektivno podešavati određene kompatibilnosti postavljanjem bitova promenljive koji njima odgovaraju. Jednom postavljeni nivo kompatibilnosti potrebno je čuvati tokom čitavog života baze.

Potrebno je podesiti registarsku promenljivu pre pravljenja baze. Takođe je važno restartovati DB2 instancu posle podešavanja vrednosti promenljive kako bi promene dobile efekat.

7.2 Kompatibilnost Orakl šeme

U okviru ovoga dela biće prikazane nove funkcionalnosti koje nudi DB2 RSUBP a tiču se podrške prilikom migracije šeme baze sa Orakl RSUBP-a na DB2 kao ciljnu platformu. Ove nove funkcionalnosti omogućavaju visok stepen automatizacije i preciznosti tokom migracije objekata šeme Orakl baze.

Prilikom migracije šeme od velikog značaja je fleksibilnost ciljnog RSUBP-a koja se ogleda u dopuštanju određenih izmena koje se tiču objekata šeme. Nove mogućnosti DB2 RSUBP-a u ovom pogledu se ogledaju u:

Nazivi objekata – U RSUBP-u Orakl nazivi objekata imaju sledeću sintaksnu formu:

```
[Naziv_šeme.]naziv_objekta[@baza]
```

Jedna od glavnih karakteristika pri organizaciji objekata u Orakl bazi ogleda se u tome što Orakl RSUBP definiše direktnu vezu između šeme i korisnika. Prostije rečeno, šema je isto što i korisnik. Samim tim korisnik koji radi sa svojim objektima nije u obavezi da prilikom navođenja imena objekata navodi i ime šeme kojoj objekat pripada.

Opcioni elemenat `[@baza]` u nazivu Orakl objekata se koristi u slučaju kada je uspostavljen link između dve Orakl baze. Ovaj pristup pokazao se kao veoma koristan prilikom rada sa objektima u testnom i produkcionom okruženju Orakl baze.

U DB2 RSUBP-u se koristi veoma sličan obrazac po kome se objektima u bazi dodeljuju imena:

```
Naziv_šeme.naziv_objekta
```

U DB2 RSUBP-u bilo koji korisnik koji ima privilegiju `IMPLICIT_SCHEMA` može da napravi objekat koji će biti dodeljen odgovarajućoj šemi.

```
CREATE TABLE šema1.tabela1 (ime char(10))
```

Izvršavanjem ove komande napravi se tabela *tabela1* koja se nalazi u prethodno definisanoj šemi sa nazivom *šema1*. Ukoliko se ne navede naziv šeme prilikom pravljenja tabele ili nekog drugog objekta baze onda se implicitno napravi nova šema sa nazivom korisnika koji tu tabelu pravi i objekat se implicitno dodeljuje toj šemi.

Tabele, pogledi, indeksi – Rad sa ovim objektima šeme u oba RSUBP-a je u osnovi isti. Razlike su veoma male i one se ogledaju u različitosti pristupa pri upravljanju ovim objektima od strane RSUBP Orakl i DB2. Na primer, definicija indeksa je direktno vezana u DB2 bazi za definiciju tabele i prilikom pravljenja tabele u DB2 bazi koristi se sledeća sintaksa:

```
CREATE TABLE tabela1 (col1 integer, col2 char(10)) in tbls1 index in tbls2
```

Iz ove naredbe možemo primetiti da će podaci iz tabele *tabela1* biti sačuvani u prostoru za čuvanje tabela *tbls1*, dok će indeksne strane biti sačuvane u prostoru za čuvanje tabela *tbls2*. Ovakav pristup je nešto drugačiji u odnosu na Orakl bazu gde se u okviru naredbe za pravljenje indeksa definiše njegov odgovarajući prostor za čuvanje indeksnih strana.

Kada je jednom definisan indeks u DB2 bazi, kasnije nije moguće izvršiti bilo koju izmenu nad njim. Ovo ograničenje se javlja zbog ne postojanja naredbe ALTER INDEX u DB2 RSUBP-u sve do verzije 9.7. Ono se prevazilazi tako što se najpre indeks izbriše, a potom se izmeni naredba za njegovo definisanje kako bi se implementirale novonastale izmene. U Orakl RSUBP-u moguće je izvršiti izmene nad bilo kojom klauzom u posebnoj naredbi za definisanje indeksa naredbom ALTER INDEX.

U RSUBP-ovima Orakl i DB2 omogućeno je da tabele, pogledi, indeksi u različitim bazama imaju iste nazive. U istoj bazi tabele i pogledi moraju imati različite nazive, dok indeksi mogu imati iste nazive kao tabele ili pogledi.

Nadograđena podrška za specifične Orakl tipove – Prilikom migracije veoma je važno preslikati odgovarajuće tipove između različitih RSUBP-ova. Ukoliko odgovarajući tipovi nisu ispravno preslikani u smislu njihove namene onda sigurno dolazi do problema u radu migriranih aplikacija koje se služe njima. Naročito je važno obezbediti podršku na ciljnom sistemu za specifične tipove izvorne baze.

Zahvaljujući nadograđenoj podršci koju pruža DB2 RSUBP moguće je definisati objekte u bazi koristeći Orakl DDL narebe bez potrebe za kasnijom izmenom strukture objekata. Da bi se postigla što veća kompatibilnost sa Orakl RSUBP-om omogućeno je i korišćenje NUMBER, NVARCHAR2 i VARCHAR2 tipa podatka kao i interpretacija datumskog tipa DATE(godina, mesec, dan) pomoću tipa TIMESTAMP(0)(godina, mesec, dan, sat, minut, sekunde). Korišćenje ovih tipova je omogućeno podešavanjem registarske promenljive DB2_COMPATIBILITY_VECTOR. U daljem tekstu će biti dat detaljniji opis podržanih Orakl osnovnih specifičnih tipova, kao i nekih složenijih.

NUMBER – podrška za tip NUMBER i NUMBER(p [,s]) u DB2 RSUBP/u bazirana je na već postojećim tipovima DECFLOAT i DECIMAL.

VARCHAR2 – za ovaj Orakl specifični tip takođe postoji podrška u RSUBP-u DB2. On omogućuje da se prazni stringovi tretiraju kao NULL vrednosti i zauzima onoliko memorijskog prostora koliko ima karaktera u stringu.

Orakl DATE tip – ovaj specifični Orakl tip sadrži pored kalendarskog vremena i dodatnu vremensku komponentu koja ukazuje na vreme izraženo u satima, minutima i sekundama.

BOOLEAN – ovaj specifični Orakl tip može biti korišćen u proceduralnoj logici, a promenljive ovog tipa su najčešće parametri procedura ili funkcija. Takođe, promenljive ovog tipa mogu biti i povratna vrednost funkcije.

VARRAY – pored podrške za regularne nizove sada postoji i podrška za asocijativne nizove koji predstavljaju kolekcije i odgovaraju složenom tipu podatka.

ROW TYPE – ovaj kompozitni tip može biti korišćen za deklaraciju promenljivih i parametara, odnosno kao elemenat nizova ili asocijativnih nizova.

REF CUR TYPE – zahvaljujući podršci za ovaj kursor promenljivu sada se kursori mogu prosleđivati između procedura i funkcija, odnosno različitih programa.

Maksimalna dužina naziva objekata šeme DB2 baze **–** Maksimalna vrednost dužine naziva objekata u bazi je povećana na 128 karaktera. Ovim je obezbeđeno da ne postoji potreba za izmenama na nazivima objekata tokom njihove konverzije.

OR REPLACE klauza u CREATE naredbi – OR REPLACE klauza specifična za Orakl CREATE naredbu predstavlja novu opciju u CREATE naredbi za pravljenje procedura, funkcija, tabela i ostalih objekata u DB2 bazi. CREATE OR REPLACE naredba je identična naredbama DROP i CREATE u navedenom redosledu. Korišćenje ove klauzule omogućava izmene na postojećem objektu ili njegovo pravljenje ukoliko

prethodno nije postojao. Ovo poboljšanje CREATE naredbe dosta olakšava posao održavanja objekata šeme u DB2 bazi. Sve privilegije nad objektom koje su važile pre njegove izmene su očuvane.

7.2.1 Konverzija sekvenci

Objekti sekvenci imaju istu definiciju u DB2 i Orakl RSUBP-u. Proces konverzije ovih objekata između Orakl i DB2 sistema ne zahteva ručne izmene u CREATE SEQUENCE naredbi.

Bitne karakteristike ovih objekata šeme koje treba napomenuti su:

- Automatski prave jedinstvene vrednosti
- Deljeni su objekti
- Mogu biti iskorišćene za pravljenje vrednosti primarnog ključa
- One prave sekvencijalne vrednosti koje se mogu koristiti u SQL naredbama
- Poboljšavaju efikasnost pristupa sekventnim vrednostima koja su privremeno sačuvana u memoriji.

Sekvenca se može koristiti od strane bilo koje aplikacije, a pristup njenim vrednostima se vrši uz pomoć izraza NEXTVAL koji vraća sledeću vrednost i CURRVAL koji vraća trenutnu vrednost sekvence.

7.2.2 Konverzija indeksa

Indeksi predstavljaju objekte šeme baze koji se koriste da bi se poboljšale performanse upita. Najvažnije karakteristike indeksa su:

- Predstavljaju objekte šeme
- Koriste se kako bi ubrzali pronalaženje željenih redova iz tabele koristeći pokazivače
- Ubrzavaju izlazno/ulazne operacije na disku na kome se nalaze podaci
- Nezavisan je od tabele nad čijom kolonom(kolonama) je postavljen
- Koriste se i održavaju automatski od strane RSUBP-a

U Orakl bazi dva tipa indeksa mogu biti napravljena. Jedan tip je UNIQUE indeks koji Orakl RSUBP automatski definiše kada nad kolonom ili nad više kolona definišemo PRIMARY KEY ili UNIQUE ograničenje. Ime takvog indeksa se poklapa sa imenom definisanog ograničenja. Drugi tip indeksa je NONUNIQUE indeks koji eksplicitno pravi korisnik.

Prethodno opisani pristup u radu sa indeksima od strane RSUBP-a Orakl veoma je sličan pristupu u DB2 bazi. Razlika se ogleda u tome što se u DB2 RSUBP –u prilikom definisanja indeksa kao dodatna opcija nudi INCLUDE klauza koja omogućuje uključivanje dodatnih kolona iz tabele u definiciju indeksa.

```
CREATE UNIQUE INDEX ix1 ON tabela
(col1 ASC) INCLUDE (col2, col3, col4, col5)
```

Zahvaljujući ovoj mogućnosti uključivanja dodatnih kolona u definiciju indeksa kao posledica se javlja poboljšanje performansi. Dodatne kolone su uključene u indeks ali za njihove vrednosti ne važi svojstvo uređenosti i jedinstvenosti.

Orakl bitmap indeksi – Za Orakl bitmap indeks ne postoji podrška u DB2 sistemu. Ovi indeksi su namenjeni za potrebe pretraživanja velikih tabela u skladištu podataka i definišu se nad kolonama koje imaju malu kardinalnost različitih vrednosti. U DB2 RSUBP-u nije moguće eksplicitno napraviti ove indekse.

Klasterovani indeksi – U Orakl i DB2 RSUBP-ovima susrećemo se sa pojmom klasterovanih indeksa. Ovaj pojam u različitim RSUBP-ovima ima različito značenje. Tako u Orakl RSUBP klasterovani indeksi se definišu nad particionisanim tabelama. U DB2 RSUBP-u klasterovani indeksi obezbeđuju klasterovanje tabele po vrednostima u njenim kolonama. Kada definišemo neki indeks sa CLUSTER opcijom onda su podaci u tabeli nad čijim kolonama je definisan ovaj indeks organizovani u skladu sa organizacijom indeksa.

7.2.3 Konverzija ograničenja

Orakl i DB2 podržavaju iste tipove ograničenja koja čuvaju integritet podataka, kao što su primarni ključ, strani ključ, unique, NOT NULL, i check ograničenja. Primarni ključ i check ograničenje su identični u obe baze. Što se tiče ostalih ograničenja, razlike u pristupu vezanom za definisanje i primenu ovih ograničenja u bazi između ova dva sistema se ogledaju u sledećem:

- Sve kolone koje su uključene u unique ograničenje u DB2 sistemu moraju biti definisane kao NOT NULL, dok Orakl dozvoljava NULL vrednosti.
- Prilikom definisanja primarnog ključa u Orakl-u nije potrebno definisati NOT NULL ograničenje nad kolonom(kolonama) koja čini primarni ključ, dok se u DB2 sistemu mora navesti NOT NULL nad kolonom(kolonama) koju definišemo kao primarni ključ.
- U DB2 sistemu je moguće definisati spoljašnji ključ i nad kolonama nad kojima je definisano UNIQUE ograničenje.

7.2.4 Konverzija pogleda

Za poglede u RSUBP-ovima DB2 i Orakl može se reći da su veoma slični i razlike među njima ogledaju se u složenosti SQL naredbi kojima su definisani. Dakle, mogu da postoje manje razlike koje se rešavaju ručno, ali u većini slučajeva pogledi koji se mogu pokrenuti na Orakl sistemu rade i na DB2 sistemu. Prilikom definisanja pogleda dozvoljeno je u DB2 sistemu koristiti Orakl SQL sintaksu CREATE OR REPLACE VIEW, sintaksu spoljašnjeg spajanja koja uključuje korišćenje znaka (+), različite tipove skalarnih funkcija, itd.

Orakl materijalizovani pogledi predstavlaju jedan od osnovnih koncepata koji se koriste za poboljšanje performansi izračunavanja nad podacima iz baze. Materijalizovani pogledi se prvenstveno koriste u oblasti skladištenja podataka i njihova uloga je da čuvaju rezultate vremenski zahtevnih upita kako se oni ne bi često izvršavali i doprinoslili trošenju dragocenog vremena. Rezultati koji se čuvaju u ovim pogledima moraju pravovremeno biti osvežavani kako bi se održala ispravnost izračunatih rezultata koji se u njima čuvaju. Materijalizovanim pogledima u DB2 sistemu odgovaraju materijalizovane upitne tabele (MQT - Materialized Query Table).

7.2.5 Konverzija sinonima

Sinonimi u Orakl RSUBP-u predstavljaju alternativna imena za objekte baze kao što su tabele, pogledi, sekvence, procedure, funkcije itd. Postoje dve vrste Orakl sinonima:

- Javni sinonimi koji su dostupni svim korisnicima
- Privatni sinonimi koji su dostupni samo u šemi trenutnog korisnika koji ih definiše

Orakl sinonimima u DB2 RSUBP-u odgovaraju aliasi. Konverzija Orakl sinonima na DB2 sistem se odvija automatski bez potrebe za ručnim intervencijama. Takođe je podržana i Orakl sintaksa za definisanje sinonima za tabele, dok za ostale objekte je potrebno naglasiti za koji tip objekta se sinonim definiše.

Sintaksa za definisanje javnog sinonima za tabelu:

```
CREATE PUBLIC SYNONYM naziv_sinonima FOR naziv_šeme.naziv_tabele
```

Sintaksa za definisanje javnog sinonima za objekat sekvence:

```
CREATE PUBLIC SYNONYM naziv_sinonima FOR SEQUENCE naziv_šeme.naziv_sekvence;
```

7.2.6 Konverzija particionisanih tabela

Particionisane tabele se prave deljenjem veoma velikih tabela u manje celine-particije, jer je jednostavnije raditi sa pojedinačnim particijama nego sa velikim tabelama. Ovakav pristup omogućava lakše održavanje tabela sa velikom količinom podataka, a takođe i pri poboljšavanju performansi upita nad velikim tabelama.

U Orakl bazi razlikujemo nekoliko strategija za particionisanje tabela koje ukazuju na koji način se podaci stavljaju u particije. Osnovne strategije su:

- Particionisanje po rangu
- Particionisanje po listi vrednosti
- Heš particionisanje

Particionisanje po rangu je veoma korisno kada postoje različiti rangovi podataka koje je potrebno sačuvati zajedno. Klasična primena ovog particionisanja jeste nad kolonama tipa date, kada je potrebno zajedno sačuvati a kasnije i obraditi podatke koji pripadaju jednom danu, mesecu itd. Istorijski podaci koji nisu više potrebni veoma lako mogu biti izbrisani iz tabele uklanjanjem particije.

```
CREATE TABLE t1 (id NUMBER, c1 DATE)
PARTITION BY RANGE (c1)
  (PARTITION t1p1 VALUES LESS THAN (TO_DATE('2007-11-01', 'YYYY-MM-DD')),
   PARTITION t1p2 VALUES LESS THAN (TO_DATE('2007-12-01', 'YYYY-MM-DD')),
   PARTITION t1p3 VALUES LESS THAN (TO_DATE('2008-01-01', 'YYYY-MM-DD')),
   PARTITION t1p4 VALUES LESS THAN (MAXVALUE)
  );
```

Pošto ne postoji mogućnost automatske konverzije particionisanih tabela između RSUBP Orakl i DB2 ovaj pristup particionisanja u Orakl-u može biti predstavljen u DB2 na sledeći način:

```
CREATE TABLE t1 (id NUMBER, c1 DATE)
PARTITION BY RANGE(c1)
(STARTING MINVALUE,
STARTING '1/11/2007' ENDING '1/1/2008' EVERY 1 MONTH,
ENDING AT MAXVALUE);
```

Particionisanje po listi vrednosti se vrši nad kolonama koje imaju diskretne vrednosti. Na ovaj način je moguće eksplicitno kontrolisati koji redovi pripadaju određenoj particiji. Prednosti ovog pristupa u particionisanju jeste što omogućava grupisanje neuređenih i nesrodnih podataka prirodnim putem.

```
CREATE TABLE tab1
(
col1 int,
col2 varchar2(2)
)
PARTITION BY LIST (col2)
(PARTITION p1 VALUES ('AB', 'MB') tablespace tbs1,
PARTITION p2 VALUES ('BC') tablespace tbs2,
PARTITION p3 VALUES ('SA') tablespace tbs3,
PARTITION p13 VALUES ('YT') tablespace tbs4,
PARTITION p14 VALUES(DEFAULT) tablespace tbs5 );
```

Odgovarajuće particionisanje u DB2 bazi bi izgledalo na sledeći način:

```
CREATE TABLE tab1
(
col1 INT,
col2 CHAR(2),
col3 GENERATED ALWAYS AS
(CASE
WHEN col2 = 'AB' THEN 1
WHEN col2 = 'BC' THEN 2
WHEN col2 = 'MB' THEN 1
WHEN col2 = 'SA' THEN 3
WHEN col2 = 'YT' THEN 4
ELSE 5
END)
)
IN tbs1, tbs2, tbs3, tbs4
PARTITION BY RANGE (col3)
(STARTING 1 ENDING 5 EVERY 1);
```

Dakle, pristup u particionisanju tabela u DB2 RSUBP-u je veoma sličan kao i u Orakl-u. U osnovi je dozvoljeno da jedna tabela bude podeljena između više fizičkih skladišta podataka preko jednog ili više prostora za čuvanje tabela. Svaki od ovih objekata odgovara jednoj particiji i dozvoljava prostorima za čuvanje tabela da sadrže skup podataka kojima se veoma lako može pristupiti.

U DB2 RSUBP-u nije podržano Orakl HASH particionisanje u kome se pomoću heš funkcije određuje koji redovi pripadaju kojim particijama. Najčešće se primenjuje nad kolonama gde nije moguće primeniti rangovsko particionisanje:

```
create table tab1 (
```

```
   col1 number(4),
   col2 varchar2(30),
   col3   number
)
partition by hash(col1) (
  partition e1 tablespace tbs1,
  partition e2 tablespace tbs2,
  partition e3 tablespace tbs3,
  partition e4 tablespace tbs4
);
```

Takođe nije moguće ručno konvertovati ni Orakl tabele nad kojima je primenjeno kompozitno particionisanje. Kod takvih tabela su particije dodatno podeljene na subparticije. Na primer kada imamo jednu tabelu particionisanu po koloni datum, najpre po mesecu a potom svaki mesec po danu. U kompozitnom particionisanju Orakl tabela razlikujemo nekoliko pristupa od kojih su najpoznatiji:

Range-Hash Partitioning – Ovaj pristup u kompozitnom particionisanju omogućava da rangovske particije budu dodatno particionisane na osnovu neke hash funkcije.

Range-List Partitioning – Ovaj pristup u kompozitnom particionisanju omogućava da rangovske particije budu dodatno particionisane na osnovu liste vrednosti.

7.3 Podrška za Orakl SQL dijalekt

Razlike između RSUBP Orakl i DB2 se ogledaju i u specifičnom SQL dijalektu, čije se strukture u smislu ključnih reči i semantike razlikuju u nekim oblastima. Takođe, za jedan proizvod karakteristična su određena svojstva SQL dijalekta koja nisu podržana u drugom. Najčešće se prilikom migracije dešava da je ograničena mogućnost rada SQL naredbi, na oba RSUBP-a, za koje su svojstvene odlike SQL dijalekta na sistemu na kome su napisane. U DB2 RSUBP-u postoji podrška za specifičnosti Orakl SQL dijalekta i u daljem tekstu će one biti predstavljene:

Orakl (+) operator spoljašnjeg spajanja tabela – Operator (+) spoljašnjeg spajanja je karakterističan za SQL naredbe napisane na Orakl RSUBP-u verzije 8i. Iako je dolaskom novih verzija Orakl proizvoda ova sintaksa odbačena i dalje postoji dosta SQL koda koji se njome služi, kao i programera koji koriste ovaj operator spoljašnjeg spajanja. Primer jednog desnog spoljašnjeg spajanja koristeći + operator:

```
SELECT A.col2, A.col3,B.col1
FROM t1 A, t2 B
WHERE A.col1 (+) = B.col2;
```

DUAL tabela – Pomoćna tabela koja se sastoji od jednog reda i jedne kolone i koja je karakteristična za Orakl RSUBP. Veoma se često koristi u aplikacijama i od strane Orakl programera. Trenutno sistemsko vreme se može dobiti pomoću DUAL tabele na sledeći način:

```
select SYSDATE from DUAL;
```

MINUS SQL operator – Skupovni operator razlike u Orakl-u. On je ekvivalent EXCEPT operatoru u DB2 koji takođe služi da pronađe razliku između dva rezultujuća skupa redova vraćenih upitima.

`TRUNCATE` naredba – Ova naredba za brzo brisanje redova iz velike tabele je karakteristična za Orakl RSUBP. Podrška za nju u DB2 RSUBP-u je od velikog značaja, posebno za ugodnost rada Orakl programera koji je veoma često koriste u svom radu.

CREATE OR REPLACE naredba – Orakl klauzula OR REPLACE omogućuje zamenu već postojećeg objekta njegovom novom verzijom. U DB2 bazi novi objekti se definišu naredbom CREATE bez OR REPLACE klauze. Dodavanjem podrške za OR REPLACE klauzu u DB2 RSUBP-u omogućeno je korišćenje ove naredbe prilikom definisanja novih objekata.

ROWNUM pseudo kolona – Orakl RSUBP koristi ROWNUM pseudo kolonu da ograniči broj redova koji će biti vraćeni SQL upitom. Korišćenje ROWNUM pseudo kolone se pokazalo kao veoma pogodno prilikom rada sa velikim tabelama. U DB2 RSUBP-u postoji ugrađena kompatibilnost za ovu pseudo kolonu. Sledi primer u kome se uz pomoć ROWNUM pseudokolone vraća prvih devet redova iz tabele:

```
select * from tab1 where ROWNUM < 10
```

Dok nije ugrađena Orakl kompatibilnost u DB2 bazi prethodni upit je se mogao dobiti uz pomoć `FETCH FIRST n ROWS ONLY klauze`:

```
select * from tab1
FETCH FIRST 9 ROWS ONLY;
```

ROWID pseudo kolona – U Orakl bazi svaki red ima svoj jedinstveni identifikator odnosno ROWID. Ova pseudo kolona sadrži fizičku adresu reda u bazi i jedinstveno ga identifikuje. Vrednosti ROWID pseudo kolone su sačuvane zajedno sa podacima u bazi i one su nepromenljive za vreme života jednog reda podataka u bazi. Korišćenjem pseudo kolone ROWID postiže se najbrži pristup jednom redu podataka u bazi. U DB2 RSUBP-u postoji ugrađena kompatibilnost za ovu pseudo kolonu.

CREATE PUBLIC SYNONYM – Predstavlja naredbu za pravljenje alternativnog naziva objekta šeme u Orakl bazi. Ukoliko je u DB2 bazi već definisan javni alias za neki objekat naredbom CREATE PUBLIC ALIAS, DB2 RSUBP dopušta definisanje istoimenog javnog sinonima za taj objekat naredbom CREATE PUBLIC SYNONIM. U trenutku kada za isti objekat postoje isti ili različiti javni sinonimi i aliasi zahvaljujući fleksibilnosti DB2 RSUBP-a ova dva alternativna naziva za isti objekat se mogu koristiti bez konflikata.

SELECT FOR UPDATE klauza – Orakl FOR UPDATE klauza u SELECT naredbi se koristi da zaključa red zahvaćen SELECT naredbom kako bi se nad njim kasnije izvršila UPDATE naredba. U DB2 RSUBP-u postoji ugrađena podrška za FOR UPDATE klauzu. Najčešće se vrši zaključavanje redova na koje ukazuje kursor:

```
CURSOR c1
IS
   SELECT col1, col2
   from tb1
   FOR UPDATE of col2;
```

Orakl notacija => za dodeljivanje parametara – Prilikom poziva procedura, funkcija Orakl RSUBP omogućava pridruživanje argumenata parametrima korišćenjem => notacije. U ovom slučaju parametri ne moraju biti poziciono dodeljeni tj. onim redosledom kojim su navedeni argumenti u definiciji

procedure ili funkcije. Ovakav pristup prilikom poziva procedura je omogućen i u RSUBP-u DB2 zahvaljujući ugrađenoj kompatibilnosti za => notaciju.

7.4 Podrška za PL/SQL u RSUBP-u DB2

U okviru ovoga dela biće prikazane odlike DB2 sistema za upravljanje bazom podataka koje pružaju podršku za Orakl PL/SQL proceduralni programski jezik. Od verzije 9.7 DB2 RSUBP pruža podršku za PL/SQL programski jezik tako što omogućuje korišćenje PL/SQL koda bez upotrebe posebnih tehnika za izmene ili njegovu translaciju u neki drugi kod. Ovo svakako olakšava posao uključivanja PL/SQL aplikacija u rad na DB2 sistemu prilikom procesa migracije, a takođe i pruža mogućnost PL/SQL programerima da upotrebe svoja stečena programerska znanja sada na novoj DB2 platformi.

U ovom delu će takođe biti prikazano na koji način se uključuje podrška za PL/SQL programski jezik u DB2 sistemu za upravljanje bazom podataka, pored toga će biti predstavljeno i na koji način se upravlja u novom okruženju sa PL/SQL objektima kao što su anonimni blokovi, procedure, funkcije, paketi i okidači.

7.4.1 Podešavanje okruženja

Da bismo uklučili podršku DB2 RSUBP-a za rad u PL/SQL programskom jeziku potrebno je podesiti vrednost registarske promenljive DB2_COMPATIBILITY_VECTOR na heksadecimalnu vrednost FF pre pravljenja baze. To postižemo izvršavanjem sledeće komande u DB2 Command Window-u:

```
db2set DB2_COMPATIBILITY_VECTOR = FF
```

Po izvršavanju prethodne komande potrebno je izvršiti resetovanje instance na kojoj će raditi nova napravljena baza. Ovaj korak je potreban da bi izvršena podešavanja dobila efekat. Kada je kompatibilnost uključena DB2 baza je spremna da prihvati definicije PL/SQL programskog jezika, odnosno osposobljena je za rad PL/SQL programa.

7.4.2 PL/SQL blokovska struktura

Ono što je karakteristično za programski jezik PL/SQL jeste njegova blokovska struktura. Osnovne programske jedinice napisane u ovom programskom jeziku predstavljaju zapravo blokove sa dodeljenim nazivom ili bez njega. Blokovi bez naziva predstavljaju anonimne blokove dok oni sa nazivom predstavljaju procedure, funkcije ili okidače.

Anonimni blokovi – predstavljaju osnovnu i najednostavniju formu PL/SQL koda. Omogućuju upotrebu proceduralne logike prilikom pravljenja koda bez njegovog čuvanja u vidu objekta u bazi. Anonimni blokovi mogu da se sastoje iz tri sekcije:

- Opcione sekcije za deklaraciju tipova i promenljivih naznačena sa ključnom rečju DECLARE.
- Obavezne izvršne sekcije koja se sastoji iz SQL i PL/SQL naredbi. Ova sekcija je naznačena sa ključnim rečima BEGIN i END.
- Opcione sekcije za praćenje izuzetaka koja je naznačena ključnom rečju EXCEPTION.

Primer jednog anonimnog bloka koji ispisuje trenutno vreme na standardnom izlazu:

```
SET SERVEROUTPUT ON;

DECLARE
   trenutno_vreme DATE := SYSDATE;
BEGIN
     dbms_output.put_line( trenutno_vreme );
END;
```

Peocedure – Procedure predstavljaju imenovane PL/SQL blokove koji prihvataju parametre i čuvaju se kao objekti šeme u bazi. Generalno, procedure se koriste kada treba izvršiti nekakvu akciju. Sastoje se iz sekcije zaglavlja, sekcije deklaracije, izvršne sekcije i opcione sekcije za praćenje izuzetaka. Procedura se poziva koristeći njen naziv iz nekog drugog PL/SQL bloka. U RSUBP-u DB2 postoji podrška za kompajliranje i izvršavanje PL/SQL procedura.

Primer procedure koja kao argument prima nisku karaktera koju ispisuje na standardni izlaz:

```
CREATE OR REPLACE PROCEDURE jednostavna_procedura (poruka varchar2(20))
IS
BEGIN
   DBMS_OUTPUT.PUT_LINE(poruka);
END jednostavna_procedura;
```

Primer poziva procedure iz anonimnog bloka:

```
BEGIN
    jednostavna_procedura('Zdravo Svete');
END;
```

Procedure se mogu pozivati korišćenjem ključne reči EXECUTE, iz drugih procedura, funkcija.

Funkcije – Funkcije predstavljaju imenovane PL/SQL blokove koji se čuvaju u bazi kao objekti šeme. One prihvataju parametre, mogu biti pozivane iz drugih blokova i obavezno vraćaju vrednost. Funkcije predstavljaju zapravo procedure koje vraćaju jednu vrednost. Kao i procedure i funkcije se sastoje iz sekcije zaglavlja, izvršne sekcije i opcione sekcije za praćenje izuzetaka. Funkcije moraju u svom kodu sadrže RETURN klauzulu u izvršnoj sekciji. U RSUBP-u DB2 postoji podrška za kompajliranje i izvršavanje PL/SQL funkcija.

Primer funkcije koja vraća nisku karaktera:

```
CREATE OR REPLACE FUNCTION jednostavna_funkcija
    RETURN VARCHAR2
IS
BEGIN
    RETURN 'Zdravo Svete';
END jednostavna_funkcija;
```

Primer poziva funkcije iz anonimnog bloka:

```
BEGIN
  DBMS_OUTPUT.PUT_LINE(jednostavna_funkcija);
END;
```

Funkcije se takođe mogu pozivati korišćenjem ključne reči EXECUTE, iz drugih procedura ili funkcija, u sklopu SQL ili PL/SQL naredbi.

PL/SQL paketi - Paket predstavlja grupu povezanih PL/SQL tipova podataka, objekata i procedura. Sastoji se od:

- Specifikacije - interfejs ka aplikaciji. Sadrži deklaracije tipova, promenljivih, konstanti, izuzetkaka, kursora i programa. Definiše se sledećom naredbom:

```
CREATE [OR REPLACE] PACKAGE <ime_paketa>
IS|AS
--Javni tipovi i deklaracija promenljivih
--Specifikacija procedura
       END <ime_paketa>;
```

- Tela paketa – sadrži pune definicije kursora i programa tj. implementaciju specifikacije. Sintaksa za definisanje tela paketa je sledeća:

```
CREATE [OR REPLACE] PACKAGE BODY <ime_paketa>
IS|AS
--privatni tipovi i deklaracija promenljivih
--telo procedura
       END <ime_paketa>;
```

Paket se ne može pozivati, ugnjezditi ili imati parametre. Format je sličan kao kod procedura. Jednom kad je napisan i kompajliran mogu ga koristiti različite aplikacije.

7.4.3 PL/SQL promenljive

Promenljive koje se koriste u blokovima generalno moraju biti definisan u sekciji za deklaraciju bloka u slučaju da se ne radi o globalnim promenljivim ili onim na nivou paketa. Sekcija za deklaraciju sadrži definicije promenljivih, kursora i drugih tipova koji mogu biti korišćeni u okviru PL/SQL naredbi u izvršnoj sekciji bloka. Deklaracija promenljive se sastoji iz dodeljivanja imena promenljivoj i tipa podatka koji odgovara toj promenljivoj. Opciono, promenljivoj se može dodeliti i predefinisana vrednost u sekciji za deklaraciju.

Funkcije i procedure mogu imati parametre za prihvatanje ulaznih vrednosti. Procedure, takođe mogu imati i parametre za prihvatanje izlaznih vrednosti, a takođe je moguće da ulazni parametar bude i izlazni i obrnuto.

DB2 pored standardnog pristupa za deklaraciju promenljivih pruža podršku i za deklarisanje tipova promenljivih koristeći atribute %TYPE, %ROWTYPE.

%TYPE atribut se koristi prilikom deklaracija promenljivih ili parametara i podržan je od strane DB2 RSUBP-a. Ovaj atribut se koristi da ukaže na kompatibilnost tipa promenljive sa već definisanim tipom neke kolone tabele u bazi ili neke druge promenljive. Ukoliko imamo definisanu sledeću tabelu:

```
tabela1 (col1 integer, col2 char(20));
```

Možemo definisati promenljivu istog tipa kao i kolona col1 tabele tabela1 na sledeći način:

```
prom1.col1%TYPE;
```

%ROWTYPE atribut se koristi prilikom deklarisanja PL/SQL promenljivih tipa slog sa poljima koja odgovaraju tipovima kolona neke tabele ili pogleda.

Slog predstavlja kolekciju polja kome je dodeljen jedinstven naziv. Polje je slično promenljivoj i ono ima naziv i dodeljeni tip, ali za razliku od promenljive ono pripada strukturi kao što je slog.

```
TYPE t1_typ IS RECORD (
  c1 T1.C1%TYPE,
  c2 VARCHAR(10)
)
```

Slog se može deklarisati i na sledeći način:

```
r_promenljiva T1%ROWTYPE;
```

Gde je promenljiva `r_promenljiva` tipa sloga čiji tipovi polja odgovaraju tipu reda tabele `T1`. Za sve pomenute PL/SQL tipove i atrubute u ovoj sekciji DB2 RSUBP daje podršku za njihovo korišćenje.

7.4.4 PL/SQL osnovne naredbe

Kao i u drugim programskim jezicima i u programskom jeziku PL/SQL postoje osnovne naredbe kao što su naredba dodele, NULL naredba, EXECUTE IMMEDIATE naredba za izvršavanje dinamičkog SQL-a, SQL naredbe za interakciju sa serverom na kome se nalazi baza.

NULL naredba ne vrši nikakvu akciju. Ona se najčešće koristi kada je ispunjen uslov koji ne pokreće nikakvu akciju.

Naredba dodele dodeljuje vrednost prethodno deklarisanim promenljivima. Ona definiše podrazmevanu vrednost promenljive koja se može menjati tokom izvršavanja programa.

EXECUTE IMMEDIATE služi za izvršavanje dinamičkog SQL-a. Ova naredba prethodno pripremljenu nisku karaktera prihvata kao SQL naredbu i izvršava je na serveru na kome se nalazi baza.

SQL naredbe DELETE, INSERT, MERGE, SELECT INTO, UPDATE se takođe mogu koristiti u izvršnoj sekciji PL/SQL bloka.

SQL%FOUND, SQL%NOTFOUND, i SQL%ROWCOUNT atributi izvršenih naredbi su takođe podržani u DB2 sistemu za upravljanje bazom podataka. Ovi atributu u PL/SQL programskom jeziku se koriste da bi se mogao prepoznati efekat neke SQL naredbe.

7.4.5 PL/SQL kontrolne naredbe

Kontrolne naredbe predstavljaju programske naredbe koje PL/SQL programskom jeziku obezbeđuju potpuno proceduralno svojstvo koje nije karakteristično za SQL jezik.

IF naredba se koristi u kontekstu izvršavanja određene SQL naredbe ukoliko je ispunjen određeni uslov. U programskom jeziku PL/SQL razlikujemo četiri forme if naredbe i to su:

- IF…THEN…END IF
- IF…THEN…ELSE…END IF
- IF…THEN…ELSE IF…END IF
- IF…THEN…ELSIF…THEN…ELSE…END IF

CASE naredba izvršava jednu ili više naredbi kada je određeni pretražujući uslov ispunjen. U programskom jeziku PL/SQL postoje dve forme CASE naredbe i to su foma jednostavne CASE naredbe i forma pretražujuće CASE naredbe. Obe forme CASE naredbe su podržane od strane RSUBP-a DB2.

Jednostavna CASE naredba povezuje izraz naveden u CASE klauzuli sa drugim izrazom koji je naveden u jednoj ili više WHEN klauza. U sledećem primeru ukoliko vrednost kolone col1 odgovara nekoj od vrednosti u WHEN klauzi onda se vrši akcija iz te klauze u smislu dodeljivanja odgovarajuće vrednosti col2.

```
CASE col1
      WHEN 10 THEN col2 := 'deset';

      WHEN 20 THEN col2 := 'dvadeset';

      ELSE col2 := 'nepoznato';
```

Pretražujuća CASE naredba koristi jedan ili više logičkih izraza kako bi odredila koja naredba treba da se izvrši.

```
CASE
      WHEN col1 = 10 THEN col2 := 'deset';

      WHEN col1 = 20 THEN col2 := 'dvadeset';

      ELSE col2 := 'nepoznato';
```

Kao i u svim drugim programskim jezicima i u PL/SQL programskom jeziku se koriste petlje kako bi se obezbedilo ponavljanje izvršavanja naredbi. U PL/SQL-u razlikujemo nekoliko različitih vrsta petlji. U DB2 RSUBP-u možemo koristiti osnovnu LOOP petlju, FOR petlju, WHILE petlju.

Osnovna LOOP petlja izvršava sekvencu naredbi PL/SQL bloka sve dok je zadovoljen uslov u EXIT klauzi. Exit klauza se može nalaziti i u FOR petlji i u WHILE petlji.

```
DECLARE

  sum INTEGER := 0;
BEGIN
LOOP
    sum := sum + 1;
       EXIT WHEN sum > 10;
  END LOOP;
```

```
END
```

FOR petlja se koristi takođe za izvršavanje naredbi više puta u okviru PL/SQL prcedure, funkcije, anonimnog bloka. Ispod je dat primer ispisivanja prvih deset brojeva na standardni izlaz uz pomoć FOR petlje.

```
BEGIN
  FOR i IN 1 .. 10 LOOP
    DBMS_OUTPUT.PUT_LINE('Iteracija # ' || i);
  END LOOP;
END;
```

WHILE petlja se koristi takođe za izvršavanje naredbi više puta. U sledećem primeru WHILE naredba u okvitu anonimnog bloka se izvršava sve dok je `sum < 11`.

```
DECLARE
  sum INTEGER := 0;
BEGIN
  WHILE sum < 11 LOOP
    sum := sum + 1;
  END LOOP;
END
```

CONTINUE naredba se najčešće koristi da filtrira podatke unutar petlje pre nego što se izvrši neka druga naredba. Dakle, ova naredba omogućava nam da prebacimo kontrolu na sledeću iteraciju unutar petlje ili da napustimo petlju.

```
BEGIN
FOR i IN 1 .. 5 LOOP
IF i = 3 THEN
CONTINUE;
END IF;
DBMS_OUTPUT.PUT_LINE('Iteracija # ' || i);
END LOOP;
END;
```

Kao izlaz iz ove petlje biće ispisani svi brojevi od jedan do pet na standardni izlazu sem trojke.

7.4.6 Obrada izuzetaka

Prilikom obrade izuzetaka potrebno je u PL/SQL izvršnom bloku uključiti dodatni opcioni EXCEPTION blok koji bi obrađivao izuzetke koji se pojavljuju tokom izvršavanja programa. Izuzetak predstavlja zapravo grešku koja se javlja tokom izvršavanja programa i koja zaustavlja izvršavanje programa. PL/SQL programski jezik kao i drugi jezici ima poseban mehanizam za praćenje i obradu pojavljenih grešaka tokom izvršavanja programa. Sintaksa EXCEPTION bloka je zapravo produžetak sintakse jednog izvršnog bloka. U Orakl bazi razlikujemo dva metoda prilikom pojavljivanja izuzetaka:

- Ukoliko se desi greška prilikom pristupa ili tokom obrade podataka na Orakl bazi, njen pridruženi izuzetak je automatski signaliziran. Na primer greška ORA-01403 se desi kada nijedan red nije vraćen SELECT naredbom i tada PL/SQL automatski definiše NO_DATA_FOUND izuzetak koji je pridružen ovoj grešci.

- U zavisnosti od toga koja je poslovna logika pokrivena PL/SQL programom, često postoji potreba za eksplicitnim signaliziranjem izuzetaka. Kontrola za eksplicitnim podizanjem izuzetaka se postiže korišćenjem RAISE naredbe. Podignuti izuzeci mogu biti ili predefinisani ili definisani od strane korisnika. Moguće je takođe i eksplicitno deklarisati izuzetke kojima bi se obuhvatlile standardne Orakl greške koje nisu već predefinisane.

Ukoliko se ne desi greška, blok jednostavno izvrši naredbu i kontrola prolazi do naredbe kojom se završava izvršni blok. Ukoliko se desi greška prilikom izvršavanja naredbe, dalja obrada u okviru naredbe je prekinuta, kontrola sada prelazi na EXCEPTION sekciju. WHEN klauza pretražuje sve izuzetke kako bi našla onaj koji odgovara nastaloj grešci. Ukoliko je izuzetak pronađen onda je odgovarajuća naredba izvršena i kontrola prelazi na naredbu kojom se završava izvršni blok. Ukoliko nije pronađen odgovarajuči izuzetak onda se zaustavlja izvršavanje programa.

Izuzetak u WHEN klauzi može biti korisnički definisan ili već predefinisan od strane sistema. Korisnički izuzeci se definišu u DECLARE sekciji bloka. Sintaksne konstrukcije PRAGMA EXCEPTION_INIT ili PRAGMA DB2_EXCEPTION_INIT se koriste neposredno posle definisanja izuzetka, i na taj način ukazuju na Orakl sqlcode ili DB2 sqlstate koji odgovaraju korisnički definisanim izuzecima.

```
DECLARE
  izuzetak1 EXCEPTION;
  izuzetak2 EXCEPTION;
  PRAGMA EXCEPTION_INIT(izuzetak2,-942);
  izuzetak3 EXCEPTION;
  PRAGMA DB2_EXCEPTION_INIT(izuzetak3,'42601');
BEGIN
  MyApp.Main(100);
EXCEPTION
  WHEN izuzetak1 THEN
      DBMS_OUTPUT.PUT_LINE('Korisnički definisan izuzetak1 je uhvaćen');
  WHEN izuzetak2 THEN
      DBMS_OUTPUT.PUT_LINE(' Korisnički definisan izuzetak2 (Nedefinisano
ime) je uhvaćen');
  WHEN izuzetak3 THEN
      DBMS_OUTPUT.PUT_LINE(' Korisnički definisan izuzetak3 (Sintaksna
greška) je uhvaćen');
END
```

Neki Orakl sistemski predefinisani izuzeci koji se mogu koristiti i na DB2 sistemu za upravljanje bazom podataka su:

NO_DATA_FOUND – nije pronađen nijedan red koji zadovoljava kriterijum SELECT klauze.
TOO_MANY_ROWS – vraćeno je više redova koji zadovoljavaju SELECT kriterijum u slučaju kada je dozvoljeno da bude vraćen samo jedan red.
INVALID_CURSOR – pokušaj pristupa skupu redova podataka kursora je nedozvoljen jer kursor nije otvoren.
INVALID_NUMBER – nevažeća numerička vrednost.
OTHERS – odgovara bilo kom izuzetku koji nije uhvaćen prethodnim izuzecima u WHEN klauzama.

LOGIN_DENIED – slučaj kada su korisničko ime ili šifra nevažeći.

RAISE_APPLICATION_ERROR je procedura koja signalizira izuzetak koji je baziran na korisnički definisanom kodu greške i poruci. Ova procedura je jedino podržana u kontekstu PL/SQL-a.
Za razliku od prethodne procedure postoji i naredba RAISE koja signalizira prethodno definisani izuzetak.

```
CREATE OR REPLACE PROCEDURE raise_demo (inval NUMBER) IS
  evenno EXCEPTION;
  oddno  EXCEPTION;
BEGIN
  IF MOD(inval, 2) = 1 THEN
    RAISE oddno;
  ELSE
    RAISE evenno;
  END IF;
EXCEPTION
  WHEN evenno THEN
    dbms_output.put_line(TO_CHAR(inval) || ' is even');
  WHEN oddno THEN
    dbms_output.put_line(TO_CHAR(inval) || ' is odd');
END raise_demo;
```

7.4.7 PL/SQL kursor promenljiva

PL/SQL kursori predstavljaju imenovanu strukturu kojoj odgovara rezultatu nekog SELECT upita. Umesto čestog izvršavanja upita može se koristiti kursor za čitanje i procesiranje rezultata upita i to jedan po jedan red u jednom trenutku vremena. U programskom jeziku PL/SQL razlikujemo dve vrste kursora:

- Statički kursori – predstavlja kursor kome je fiksno pridružen rezultat nekog upita.
- Kursor promenljive – predstavlja kursor koji sadrži pokazivač na određeni rezultujući skup određenog upita.

U DB2 sistemu možemo koristiti sve naredbe koje su usko povezane u radu sa kursorima. To su sledeće osnovne naredba za rad sa kursorima:

OPEN - rezultujući skup koji odgovara nekom kursoru ne može se koristiti u radu ukoliko kursor nije otvoren naredbom OPEN.

FETCH – naredba koja omogućuje pristup jednom redu u rezultujućem skupu koji odgovara nekom kursoru.

CLOSE – naredba kojom se zatvara kursor. Posle ove naredbe više nije moguće pristupati redovima iz rezultujućeg skupa koji odgovara kursoru.

Svaki kursor ima skup atributa koji omogućuje programu koji radi sa kursorom da testira stanje u kome se kursor trenutno nalazi. Rad sa ovim atributima je podržan u DB2 bazi i u njih spadaju:

%ISOPEN – atribut uz pomoć koga se proverava da li je kursor otvoren.

%FOUND – atribut uz pomoć koga se proverava da li kursor sadrži redove posle izvršenja FETCH naredbe. Suprotna svojstva njemu ima atribut %NOTFOUND.

%ROWCOUNT – atribut koji ukazuje koliko redova ima u rezultujućem skupu koji odgovara nekom kursoru.

DB2 baza podržava deklaraciju kursor promenljive koja je tipa REF CURSOR i koja se koristi prilikom prosleđivanja rezultujućeg skupa podataka između rutina u bazi ili različitih klijentskih aplikacija. Kursor promenljive su dinamičke i nisu vezane za specifični upit za razliku od kursora koji su statički.

7.4.8 PL/SQL kolekcije

PL/SQL kolekcije predstavljaju skup uređenih elemenata istog tipa. Razlikujemo tri tipa PL/SQL kolekcija i to su: asocijativni nizovi, ugnježdene tabele i jednodimenzioni tip VARRAY. Rad sa ugnježdenim tabelama nije podržan u DB2 sistemu dok za preostala dva tipa postoji podrška.

Asocijativni nizovi predstavljaju skup ključ – vrednost parova, gde je svaki ključ jedinstven i uz pomoć njega se pronalazi odgovarajuća vrednost u nizu. Ključ može biti ili celobrojna vrednost ili karakter. Vrednosti mogu biti sklarnog tipa (jedna vrednost) ili skup slogova. PL/SQL sintaksa za definisanje asocijativnih nizova je sledeća:

```
TYPE <naziv_asoc_niza> IS TABLE OF <tip_elemenata> INDEX BY
INTEGER|BINARY_INTEGER|PLS_INTEGER|VARCHAR2(veličina);
```

Jednodimenzioni tip VARRAY predstavlja niz promenljive dužine koji takođe čini kolekciju homogenih elemenata. PL/SQL sintaksa za definisanje nizova promenljive dužine je sledeća:

```
TYPE <varray tip> IS VARRAY( <max_vrednost_indeksa> ) OF <tip podatka>;
```

Ugnježdene tabele sadrže skup vrednosti. Drugim rečima, one predstavljaju tabele unutar tabela baze. Veličina ugnježdenih tabela je neogeraničena i ona se može povećavati dinamički. Mogu se koristiti i kao PL/SQL tipovi i kao objekti baze. Kao PL/SQL tipovi, ugnježdene tabele su zapravo jednodimenzioni nizovi čija se veličina dinamički povećava.

Metodi kolekcija se mogu koristiti pri dobijanju određenih vrednosti sačuvanih u kolekcijama ili pri modifikaciji kolekcija. Tu razlikujemo nekoliko osnovnih metoda koje se najčešće koriste:

- COUNT – vraća broj elemenata kolekcije.
- DELETE – briše sve elemente iz kolekcije.
- FIRST – vraća najmanju vrednost indeksa u kolekciji.
- EXTEND – dodaje jedan NULL element kolekciji.

Prethodno navedeni, kao i svi ostali metodi koji se koriste u radu sa kolekcijama su podržani u DB2 sistemu za upravljanje bazom podataka.

7.4.9 PL/SQL okidači

Triger je uskladišteni PL/SQL blok koji se izvršava kad se odigra određeni događaj. Događaj može biti povezan bilo sa tabelom, pogledom, šemom ili bazom podataka. Za razliku od procedura i funkcija, koje je potrebno ekspicitno pozivati, okidači se izvršavaju automatski (implicitno se pozivaju) kad god se desi događaj za koji je definisan dati okidač.
U događaje koji pokreću izvršavanje nekog okidača spadaju:

- Neka od DML naredbi (INSERT, UPDATE ili DELETE)
- Neka od DDL naredbi (CREATE, ALTER ili DROP)
- Operacija nad bazom (SERVERERROR, LOGON, LOGOFF, STARTUP ili SHUTDOWN)

DML triger se poziva kada se izvrši neka DML naredba, DDL triger se poziva kada se izvrši neka DDL naredba itd. Sistemski trigeri se definišu za šemu ili bazu podataka. Triger definisan za šemu se poziva za svaki događaj koji je povezan sa vlasnikom šeme (trenutni korisnik). Triger definisan nad bazom podataka se poziva za svaki događaj koji je povezan sa svim korisnicima. Triger se automatski poziva kad se bilo koji od ovih događaja desi. Ograničenje na trigeru specificira logički izraz koji mora biti zadovoljen da bi se događaj desio. Uslov se zadaje upotrebom WHEN klauzule.

Jednostavan DML okidač se može pozvati tačno u jednom vremenskom trenutku: pre (BEFORE) ili posle (AFTER) u odnosu na događaj.

DB2 sistem podržava i okidače na nivou jednog reda (ROW okidači) i okidače na nivou jedne naredbe.

- Row okidač se poziva jednom za svaki red na koji ima uticaja. Koristi FOR EACH ROW klauzulu. Korisni su ako akcija okidača zavisi od broja redova na koji utiče.
- Okidači na naredbe se pozivaju samo jednom bez obzira na koliko redova tabele imaju uticaja.

Takođe je podržan i rad sa specijalnim promenljivim PL/SQL okidača, NEW i OLD. Ove promenljive se mogu koristiti u okidačima bez njihovog eksplicitnog definisanja.
U PL/SQL okidačima se mogu koristiti i predikati UPDATING, DELETING i INSERTING koji prepoznaju naredbu koja je pokrenula okidač na izvršavanje.

8. Kontrola konkurentnog rada u RSUBP Orakl i DB2

Gledano unazad jedna od glavnih razlika između RSUBP Orakl i DB2 ogleda se u kontroli konkurentnog rada nad bazom. Osnovni princip pri kontroli konkurentnog rada od strane više korisnika nad bazom, RSUBP Orakl zasniva na konceptu "čitači ne blokiraju one koji pišu, pisači ne blokiraju one koji čitaju". Dakle blokada sloga postoji od strane pisača koji nad njim pišu, u trenutku kada drugi pisači pokušavaju da pristupe istom slogu.

U RSUBP-u DB2 je implementiran osnovni CS (Cursor Stability) nivo izolovanosti koji tradicionalno u kontroli konkurentnog rada nad bazom imao pristup: "pisači blokiraju čitače i u nekim slučajevima čitači mogu blokirati pisače". Razlog u ovakvom pristupu je se ogledao u tome što se nijedan podatak promenjen od strane trenutne transakcije ne može čitati od strane druge njoj istovremene transakcije sve dok promene nisu potvrđene. Ovaj nivo izolovanosti obezbeđuje samo da konkurentne transakcije ne mogu menjati trenutno aktivne redove otvorenih kursora posmatrane transakcije.

Tradicionalni DB2 pristup u konkurentnom radu može biti prikazan sledećom tabelom:

Trenutna transakcija	Ponašanje	Nova transakcija
Čitac	Bez blokiranja	Čitač
Čitač	Ponekad blokira	Pisač
Pisač	Blokira	Čitač
Pisač	Blokira	Pisač

Različiti pristupi u kontroli konkurentnosti od strane različitih RSUBP, prilikom migracije aplikacija se negativno odražavaju na njihovo funkcionisanje na novom sistemu. Inženjeri IBM-a su prilikom definisanja nivoa kompatibilnosti RSUBP-a DB2 verzije 9.7 uzeli u obzir stari pristup u konkurentnosti koji su izmenili i potpuno ga prilagodili Orakl pristupu. Ta izmena se zasniva na činjenici da ne postoji razlog zbog kog bi konkurentna transakcija koja radi pod CS nivoom izolovanosti čekala potvrdu druge transakcije koja vrši izmene nad zajedničkim slogom. Jednako zadovoljavajuće ponašanje bi bilo da konkurentna transakcija pročita najkasnije potvrđenu verziju podatka u slogu. Ovaj podatak može pročitati iz log bafera. Pored ove izmene onemogućeno je pravo zaključavanja čitaocima onih slogova kojima imaju pristup.

Posle izvršenih izmena ponašanje RSUBP-a DB2 u konkurentnom radu korisnika nad podacima je identično ponašanju Orakl RSUBP-a is ono se može predstaviti sledećom tabelom:

Trenutna transakcija	Ponašanje	Nova transakcija
Čitac	Bez blokiranja	Čitač
Čitač	Bez blokiranja	Pisač

Pisač	Bez blokiranja	Čitač
Pisač	Blokira	Pisač

Zahvaljujući ovom nivou kompatibilnosti u konkurentnom pristupu RSUBP-a DB2 u odnosu na RSUBP Orakl mogućnost neispravnog rada Orakl aplikacija na DB2 RSUBP-u je svedena na minimum.

9. Praktični deo migracije Orakl baze na DB2 RSUBP

U okviru ovoga dela će biti predstavljen praktični deo migracije Orakl baze na DB2 RSUBP uz pomoć alata za migraciju IBM Data Movement Tool. Najpre su opisani pripremni poslovi koje je potrebno odraditi kako bi se podesilo okruženje što predstavlja preduslov za započinjanje jednog procesa migracije ovim pristupom. U te pripremne poslove spadaju: podešavanje okruženja, podešavanje alata za migraciju, pravljenje ciljne DB2 baze i podešavanje njene Orakl kompatibilnosti i na kraju, pred sam početak procesa migracije, se izvršava planiranje i procena poslova migracije. U sledećem delu je dat opis poslova migracije koji se izvršavaju uz pomoć alata za migraciju. Na kraju je dat pregled određenih nekompatibilnosti koji su se pojavili u toku procesa migracije prilikom konverzije pojedinih objekata šema Orakl baze i PL/SQL koda.

U ovom delu kao primeri šema za konverziju su korišćene Orakl predefinisane šeme: HR, OE, SH. Ove šeme, tačnije definicije njihovih objekata se dobijaju sa instalacijom Orakl baze. Sve zajedno opisuju tri različita sektora jedne kompanije.

HR šema (Human Resources) – šema koja odgovara odeljenju za ljudske resurse. Sadrži podatke o zaposlenim i organizaciji kompanije.

OE šema (Order Entry) – u ovoj šemi se čuvaju sve informacije koje se tiču inventara proizvoda i različitih prodajnih kanala.

SH šema (Sales History) – u ovoj šemi se čuva poslovna statistika zahvaljujući kojoj je potrebno donositi pametne poslovne odluke.

Svi objekti definisani u ovim šemama i upotrebljeni PL/SQL kod su iskorišćeni kao primeri za konverziju između Orakl i DB2 sistema za upravljanje bazom podataka.

9.1 Podešavanje okruženja za migraciju na DB2 RSUBP-u

Da bismo uspešno izvršili migraciju Orakl baze na DB2 RSUBP potrebno je podesiti okruženje u okviru koga se odigrava proces migracije. Potrebno je prvo umrežiti računar na kome se nalazi instaliran RSUBP DB2 sa računarom na kome se nalazi Orakl baza. Alat za migraciju (IBM Data Movement Tool) se instalira na računar na kome se nalazi DB2 RSUBP, koji predstavlja ciljni sistem u procesu migracije. Pošto je IDMT (IBM Data Movement Tool) alat baziran na javinoj tehnologiji potrebno je da na ciljnom računaru bude instalirana i java virtuelna mašina koja će omogućiti rad alata za migraciju.

9.2 Podešavanje alata za migraciju

IBM Data Movement Tool predstavlja visoko skalabilan alat baziran na Java platformi. Da bi ovaj alat mogao da uspostavi konekciju sa bazama između kojih se vrši migracija potrebno je obezbediti drajvere koji pružaju neophodan interfejs za uspostavljanje konekcije:

- za Orakl bazu to su drajveri: *ojdbc5.jar ili ojdbc6.jar ili ojdbc14.jar, xdb.jar, xmlparserv2.jar ili classes12.jar*
- za DB2 bazu to su drajveri: *db2jcc.jar, db2jcc_license_cu.jar ili db2jcc4.jar, db2jcc4_license_cu.jar*

Takođe je potrebno podesiti radni direktorijum koji će poslužiti kao repozitorijum u kome će se čuvati svi napravljeni skriptovi za migraciju objekata i podataka od strane alata za miraciju.

9.3 Pravljenje DB2 baze i podešavanje kompatibilnosti

Kako je konekcija na ciljnu bazu neophodna za rad alata za migraciju, u prvom koraku je potrebno napraviti bazu na DB2 RSUBP-u. Na DB2 RSUBP-u preporučeno je prilikom pravljenja baze definisati automatsko upravljanje pristupom disku i veličinu strane od 32 kb. Da bi proces migracije mogao da počne, neophodno je da DB2 instanca i baza rade u Orakl kompatibilnom pežimu. Prethodna podešavanja se postižu pokretanjem sledećeg niza komandi iz DB2 Comand Window-a:

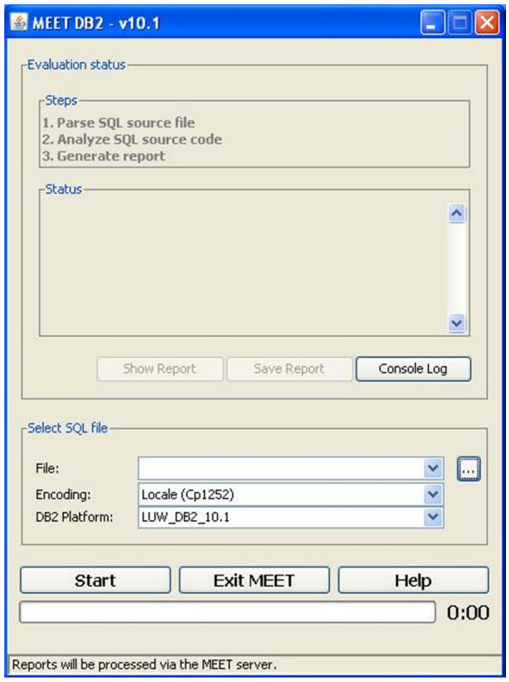
```
C:\> db2set DB2_COMPATIBILITY_VECTOR=ORA
C:\> db2stop force
C:\> db2start
C:\> db2 "create db testdb automatic storage yes on C: PAGESIZE 32 K"
```

Posle podešavanja kompatibilnosti DB2 baze sa Orakl bazom, postavljanjem registarske promenljive `DB2_COMPATIBILITY_VECTOR` na vrednost `ORA`, potrebno je restartovati DB2 bazu kako bi izvršene promene dobile efekat.

9.4 Planiranje i procena poslova migracije pomoću IBM MEET DB2 alata

Da bi se proces migracije objekata u DB2 bazu izvršio u unapred predviđenom vremenskom roku, potrebno je pre započinjanja samog procesa izvršiti procenu složenosti i brojnosti objekata Orakl baze koje je potrebno migrirati. Ovaj veoma važan korak u projektu migracije može se odraditi uz pomoć IBM MEET DB2 (Migration Evaluation and Enablement Tool) alata. Ovaj alat vrši procenu datoteka u kojima su izdvojeni objekti Orakl baze (tabele, pogledi, okidači, procedure, funkcije) kako bi odredio nivo kompatibilnosti sa DB2 bazom. Izveštaj napravljen od strane alata je u HTML formatu i on ukazuje na nivo kompatibilnosti ekstrahovanih objekata. Takođe daje i preporuke koje ukazuju na naredbe koje nisu kompatibilne i kod kojih je potrebno izvršiti intervenciju.

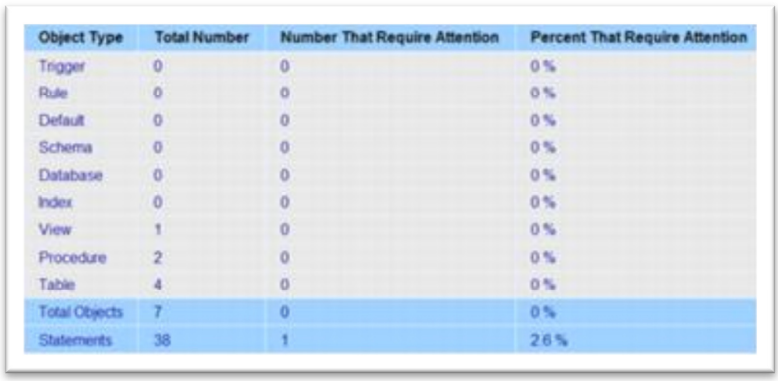
Glavna konzola alata je veoma intuitivna i laka za upotrebu. Potrebno je ukazati na ulaznu datoteku, zatim podesiti o kojoj se izvornoj bazi radi i podesiti tip kodiranja.



Izveštaj koji definiše alat se sastoji iz sledeća tri dela:

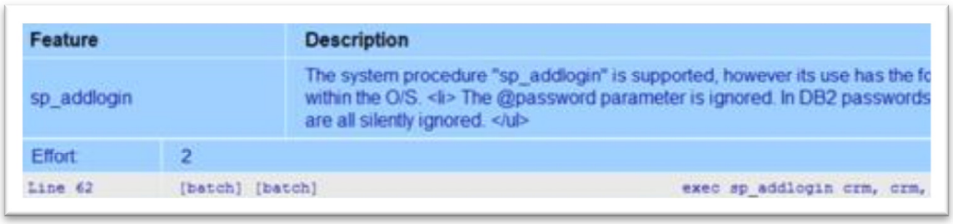
- Na sledećoj slici je prikazan izvršni rezime koji ukazuje na nivo kompatibilnosti naredbi i objekata.

- Sledeći deo izveštaja daje spisak svih objekata koji treba da migriraju na DB2 bazu i ocenu njihove kompatibilnosti.



Object Type	Total Number	Number That Require Attention	Percent That Require Attention
Trigger	0	0	0 %
Rule	0	0	0 %
Default	0	0	0 %
Schema	0	0	0 %
Database	0	0	0 %
Index	0	0	0 %
View	1	0	0 %
Procedure	2	0	0 %
Table	4	0	0 %
Total Objects	7	0	0 %
Statements	38	1	2.6 %

- Ovaj deo izveštaja omogućava analizu tako što daje statistiku za sve naredbe u ulaznim datotekama. Takođe daje prikaz nepodržanih naredbi i ukazuje na korake u cilju izmene tih naredbi. Veoma je koristan deo ovog izveštaja iz razloga što ukazuje na količinu napora koju je potrebno uložiti na izvršavanje ukazanih rešenja za nekompatibilnosti.



Feature	Description
sp_addlogin	The system procedure "sp_addlogin" is supported, however its use has the fo within the O/S. The @password parameter is ignored. In DB2 passwords are all silently ignored.
Effort	2
Line 62	[batch] [batch] exec sp_addlogin crm, crm,

9.5 Izdvajanje podataka i objekata iz izvorne Orakl baze

Pre pokretanja alata potrebno je imati sledeće informacije kako bi se uspostavila konekcija na ciljnu i izvornu bazu:

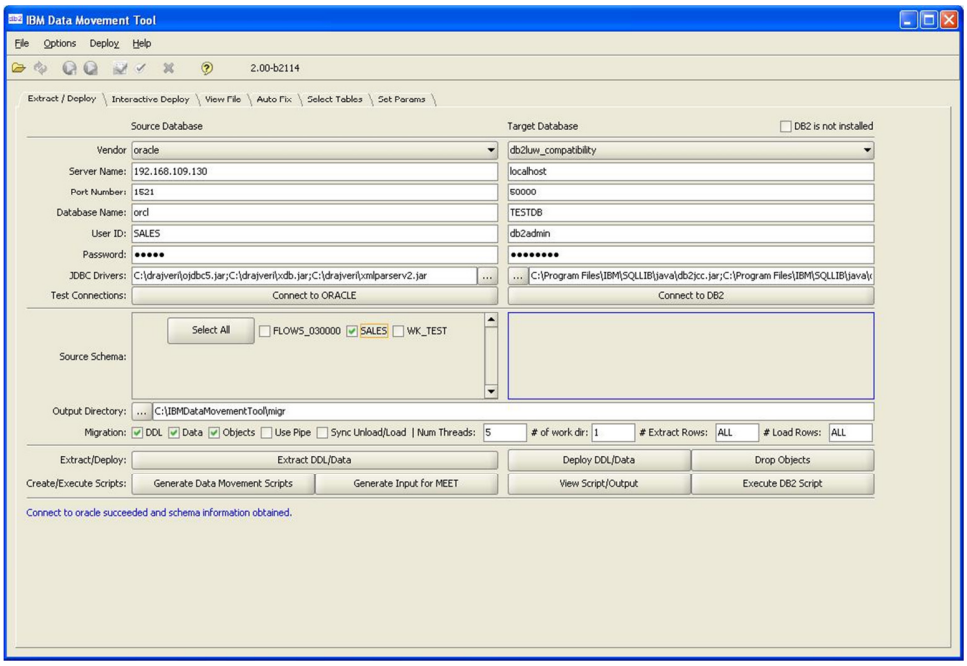
- IP adresa i korisničko ime za izvornu i ciljnu bazu
- Redni brojevi portova preko kojih se uspostavlja konekcija
- Nazivi ciljne i izvorne baze
- Nazivi korisnika sa administratorskim privilegijama
- Lozinke korisnika
- Potrebni drajveri za uspostavljenje konekcije
- Dovoljno memorijskog prostora za podatke koji migriraju na ciljni sistem

IDMT jeste besplatan alat koji se može skinuti sa interenta. Njegova instalacija je trivijalna i vrši se tako što se samo raspakuje paket u kome se program nalazi. Alat se na Windows platformi pokreće dvostrukim klikom na skript IBMDataMovementTool.cmd.

Posle pokretanja alata pojavljuje se GUI prozor u koji je potrebno uneti prethodno navedene parametre kako bi se uspostavila konekcija sa ovog alata ka izvornoj I ciljnoj bazi. Ukoliko nije podešena kompatibilnost DB2 baze onda se javlja poruka o grešci.

9.5.1 Korišćenje grafičkog korisničkog interfejsa alata za migraciju

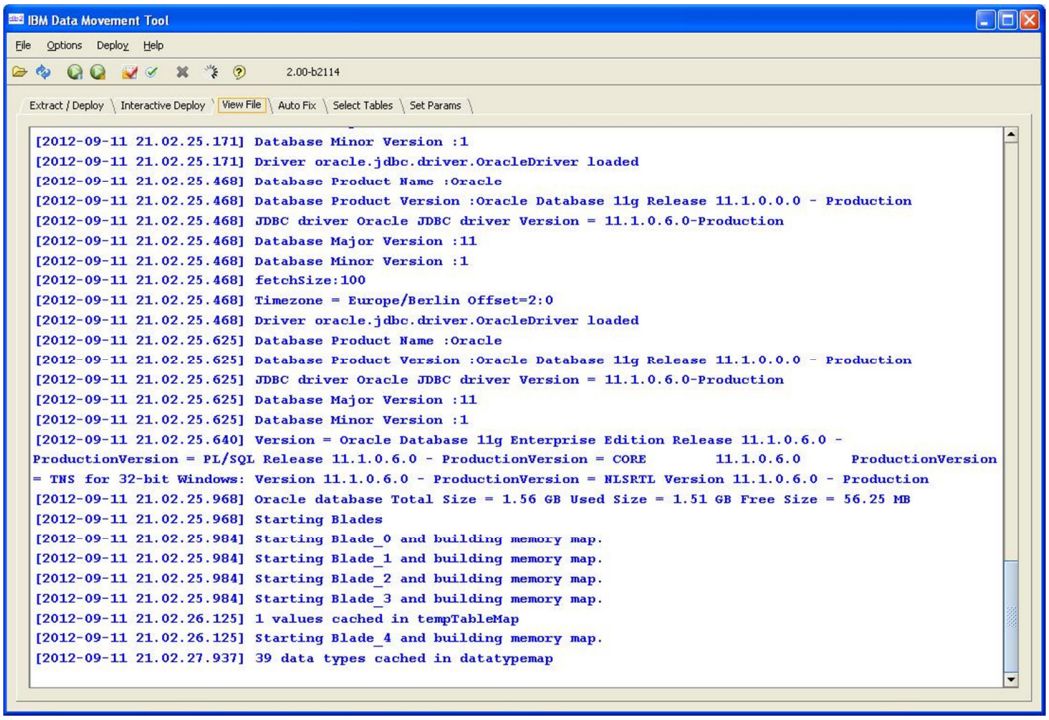
Korisnički interfejs alata je dosta intuitivan i lak za upotrebu. Posle pokretanja aplikacije javlja se prozor na kome je potrebno odabrati karticu Extract/Deploy.



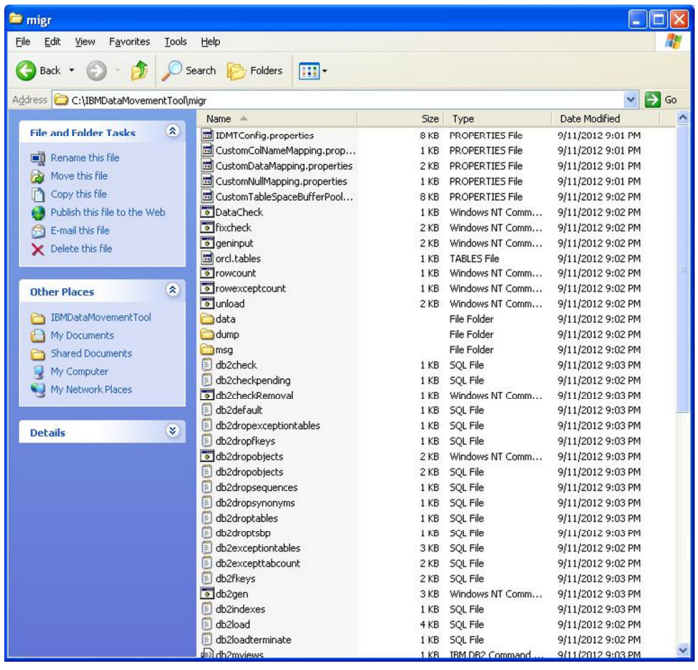
Da bi se uspostavile konekcije preko alata potrebno je navesti sve prethodno prikupljene informacije o konekcijama kao i lokacije na kojima se nalaze potrebni drajveri u odgovarajuća polja. Proveru konekcija vršimo pomoću kontrola za testiranje konekcija. Pokretanjem ovih kontrola dobijamo izveštaj da li je konekcija uspešno uspostavljena, u suprotnom dobijamo poruku o grešci.

Kada su konekcije uspešno uspostavljene, sledeći korak je specificiranje direktorijuma na fizičkom disku gde će alat sačuvati sve skriptove koje automatski napravi. Ovaj direktorijum predstavlja zapravo radni direktorijum.

Kada su konekcije uspostavljene, sledeći korak je izdvajanje DDL naredbi i podataka iz Orakl baze. Ovaj proces započinjemo klikom na kontrolno dugme Extract DDL/Data. Posebna pogodnost koju nudi alat jeste mogućnost interaktivnog praćenja procesa izdvajanja. To je omogućeno pomoću kartice View File.



Na kraju procesa ekstrakcije svi napravljeni skriptovi i datoteke sa podacima se nalaze u radnom direktorijumu. Neke od ovih kontrolnih datoteka se mogu pokrenuti i ručno iz komandne linije na DB2 RSUBP-u.



Medju datotekama u radnom direktorijumu koje je napravio alat za migraciju nalaze se i datoteke koje su usko povezane sa poslovima migracije podataka između dve baze. To su:

IBMExtract.properties – Ova datoteka sadrži sve ulazne parametre koji su specificirani kroz grafički korisnički interfejs. Moguće je promeniti sadržaj ove datoteke ručno kako bi se izmenile vrednosti parametara. Ova datoteka je ponovno napravljena svaki put kada se pokreće grafički korisnički interfejs alata za migraciju.

Unload – Ovaj skript je takođe napravljen od strane alata. Zahvaljujući njemu se izdvajaju svi podaci u nestruktuirane datoteke ukoliko je odabrana DDL /Data opcija prilikom definisanja plana izdvajanja. Takođe je moguće da ovaj skript prebacuje podatke između izvorne i ciljne baze koristeći tehniku pajpova ukoliko je ta opcija definisana prilikom pravljenje plana migracije.

Rowcount – Ovaj skript može poslužiti, posle izvršenog procesa migracije podataka, za verifikaciju broja redova podataka u tabelama ciljne baze u odnosu na tabele izvorne baze.

Svi prethodno opisani skriptovi se mogu iskoristiti za izvršavanje poslova migracije podataka ručno, dakle ne uz pomoć korisničkog interfejsa alata. Ovaj pristup je naročitio pogodan kada se radi o velikoj količini podataka i obekata baze, gde se otvara mogućnost automatizacije migracije podataka kroz “batch” proces.

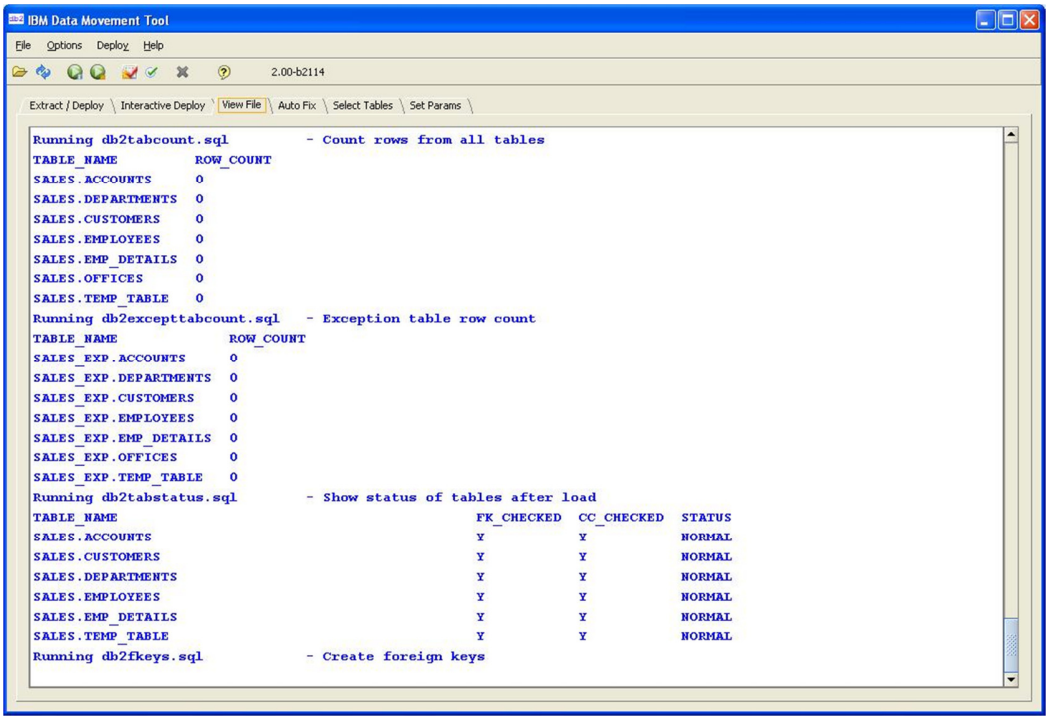
9.6 Učitavanje objekata i podataka u DB2 bazu

Posle izdvajanja objekata i podataka izvorne baze sledeći korak je njihovo učitavanje u ciljnu DB2 bazu koje se može uraditi na tri različita načina:

- Upotrebom kontrolne opcije *Deploy DDL/DATA* čime pokrećemo proces učitavanja DDL naredbi i učitavanja podataka.
- Korišćenjem *Interactiv Deploy* kartice koja pruža mogućnost interaktivnog učitavanja objekata.
- Učitavanje objekata i podataka iz komandne linije korišćenjem napravljenog skripta *db2gen* od strane alata prilikom procesa izdvajanja.

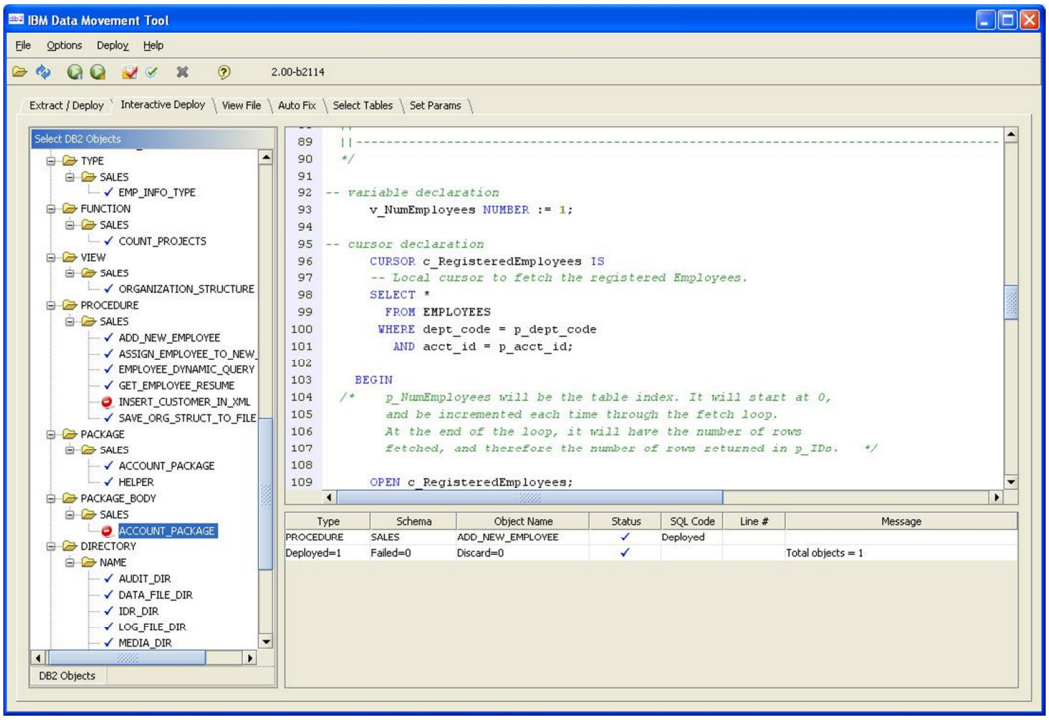
Koji od prethodna tri načina će se koristiti zavisi od toga kakvi su zahtevi prilikom migracije objekata i podataka. Ukoliko se migriraju samo oni objekti koji nisu PL/SQL objekti i podaci onda se taj proces može jednostavno izvršiti korišćenjem kontrolne opcije *Deploy DDL/DATA* ili pokretanjem skripta *db2gen*. Migraciju PL/SQL objekata je najpogodnije raditi interaktivno. Ovaj pristup omogućuje migriranje korak po korak delova PL/SQL koda čime se ima uvid u sve moguće nekompatibilnosti koda koje se javljaju prilikom migracije.

Kad je završen proces učitavanja objekata i podataka, koji je moguće pratiti preko *View File* kartice, dobija se poruka o uspešnom završetku posla.



Kada su podaci i objekti uspešno migrirani na novu platformu, sledi korak u kome se migriraju preostali PL/SQL objekti. Ovaj posao se obavlja preko prozora za interaktivno učitavanje u nekoliko koraka:

- Otvoriti prozor za interaktivno učitavanje.
- Podesiti radni direktorijum u kome se nalaze izdvojeni PL/SQL objekti, preko kontrolne opcije *Open Directory***.** Ovim postižemo da su objekti, čije su definicije izdvojene u datotekama, izlistani u obliku drveta.
- Moguće je izvršiti učitavanje svih objekata odjednom korišćenjem kontrolnog dugmeta *Deploy ALL Objects*. U okviru ovog procesa neki objekti će migrirati a kod nekih će se javiti greška.
- Za one objekte koji nisu uspešno učitani u delu za editovanje je moguće videti njihov izvorni kod, a takođe se pruža mogućnost uvida i u grešku koja je se javila.
- Takođe je moguće vršiti učitavanje jednog ili grupe objekata korišćenjem kontrolne opcije *Deploy Selected Objects*.



Iako postoji mogućnost rada DB2 baze u Orakl režimu i dalje se prilikom migracije dešava da postoji nekompatibilnost, najčešće PL/SQL objekata. Ove nekompatibilnosti se mogu rešiti pri samom procesu učitavanja PL/SQL objekata zahvaljujući alatu za migraciju koji pruža interfejs za menjanje i ukazivanje na greške koje se pojavljuju pri procesu migracije.

9.7 Orakl nekompatibilnosti DB2 baze

I pored ugrađenog nivoa kompatibilnosti za objekte šeme Orakl baze i PL/SQL programe u DB2 RSUBP-u postoje i dalje određene razlike pri radu sa određenim objektima šeme i nekompatibilnosti PL/SQL koda. U daljem tekstu će biti dat prikaz ovih nekompatibilnosti koje se javljaju kao posledica različitog pristupa u radu sa PL/SQL kodom i određenim objektima šeme od strane RSUBP Orakl i DB2.

Pravljenje i definisanje tipova – DB2 baza daje podršku za definisanje korisničkih tipova. Međutim, razlika koja je očigledna u odnosu na Orakl bazu ogleda se u tome što kada definišemo korisničke tipove u DB2 bazi onda to činimo u okviru nekog PL/SQL paketa. Korisničke tipove nije moguće definisati u telu neke PL/SQL procedure ili funkcije kao što je to slučaj kada radimo sa PL/SQL programskim jezikom nad Orakl bazom. Ukoliko želimo da se koristimo u kodu neke procedure sa prethodno definisanim tipom u nekom paketu onda ga referenciramo uz pomoć naziva paketa.

Orakl OBJECT tip - Struktuirani tipovi su podržani i konceptualno veoma slični u Orakl i DB2 sistemima za upravljanje bazom podataka. Orakl struktuirani OBJECT tip jeste zapravo korisnički definisan tip koji se može sastojati od polja atributa i specifikacija metoda. Primer naredbe za definisanje jednog Orakl OBJECT tipa:

```
CREATE OR REPLACE TYPE naziv_tipa AS OBJECT (
  Atribut1      VARCHAR2(30),
```

```
  Atribut2      CHAR(15),
  Atribut3      VARCHAR2(30),
  Atribut4      VARCHAR2(10),
  MEMBER FUNCTION funk1 (param1 tip_parametra) RETURN povratna_vrednost,
  MEMBER FUNCTION funk2 (param2 tip_parametra) RETURN povratna_vrednost
);
```

I pored ugrađenog nivoa kompatibilnosti za Orakl struktuirane tipove u DB2 bazi nije moguće konvertovati prethodno definisani OBJECT tip. Moguća je samo konverzija OBJECT tipa koji u svojoj definiciji ima spisak atributa, pri čemu se naredba `CREATE OR REPLACE TYPE naziv_tipa AS OBJECT` zamenjuje naredbom `CREATE TYPE naziv_tipa AS ROW.`

Ručna konverzija prethodno definisanog OBJECT tipa u DB2 bazi bi bila:

```
CREATE TYPE naziv_tipa AS
  (Atribut1     VARCHAR2(30),
   Atribut2     CHAR(15),
   Atribut3     VARCHAR2(30),
   Atribut4     VARCHAR2(10))
   NOT FINAL
   MODE DB2SQL

     METHOD funk1 (param1 tip_parametra)
     RETURNS povratna_vrednost
     LANGUAGE SQL
     DETERMINISTIC
     CONTAINS SQL
     NO EXTERNAL ACTION,
     METHOD funk2 (param2 tip_parametra)
     RETURNS povratna_vrednost
     LANGUAGE C
     DETERMINISTIC
     PARAMETER STYLE SQL
     NO SQL
     NO EXTERNAL ACTION
```

CREATE DIRECTORY – Orakl direktorijum predstavlja objekat baze koji je zapravo neka vrsta alijasa za direktorijum u kome su locirani podaci za eksterne tabele i eksterne velike binarne datoteke LOB. Prilikom migracije skripta za pravljenje direktorijum objekta javila se greška koja se odnoci na CREATE DIRECTORY naredbu. Ona je zamenjena sa odgovarajućom DB2 naredbom UTL_DIR.CREATE_DIRECTORY.

WITH READ ONLY – Ova klauza se koristi u Orakl bazi samo onda kada želi da se naglasi u definiciji pogleda da su podaci u njemu dostupni samo za čitanje. Primer naredbe za definisanje jednog takvog pogleda u DB2 bazi bi bio:

```
CREATE VIEW pogled1 AS SELECT * FROM tabela1 WITH READ ONLY;
```

Implementacija prethodnog u DB2 bazi bi bila:

```
CREATE VIEW pogled1 AS SELECT tabela1.* FROM tabela1, DUAL;
```

ORDER BY klauzula u definiciji pogleda – U DB2 bazi postoji ograničenje na korišćenje ove klauzule od strane glavne SELECT naredbe u definiciji pogleda. Razlog se ogleda u tome što ne postoji garancija da će

ova klauza biti uzeta u obzir prilikom izvršavanja SELECT upita nad pogledom. S druge strane, u Orakl bazi ne postoji ovakva vrsta restrikcije I moguće je dobiti uređen skup podataka prilikom pretraživanja pogleda u čijoj definiciji se koristi ORDER BY klauza. Primer jedne takve definicije u Orakl bazi:

```
CREATE VIEW pogled AS SELECT * FROM tabela ORDER BY col1;
```

Prethodna definicija pogleda se može prepisati u DB2 bazu na sledeći način:

```
CREATE VIEW pogled AS SELECT * FROM (SELECT * FROM tabela ORDER BY col1);
```

U ovakvom pristupu definisanja pogleda ove vrste u DB2 bazi ne postoji gubitak performansi.

Orakl privremene tabele – U DB2 bazi postoji podrška za privremene tabele koje se definišu pomoću DECLARE GLOBAL TEMPORARY TABLE naredbe. Jednom definisana DGGT (Declared Global Temporary Table) vidljiva je samo trenutnoj sesiji i ne može biti deljena između različitih sesija. Takođe, njena definicija se ne pojavljuje u sistemskom katalogu. Svaka sesija koja definiše DGGT poseduje svoju jedinstvenu definiciju privremene tabele. Kada se sesija završi, svi redovi podataka iz privremene tabele se obrišu kao i sama definicija privremene tabele.

Koncept privremenih tabela se znatno razlikuje u Orakl bazi. Sesija jedne aplikacije ili korisnika može da koristi privremenu tabelu u Orakl bazi za čuvanje podataka bez preplitanja u radu sa drugim aplikacijama ili korisnicima koji je u istom trenutku koriste. Drugim rečima rečeno, definicija privremene tabele je sačuvana u bazi i deljena između svih konkurentnih aplikacija ili korisnika koji rade nad bazom. Prilikom pokretanju nove sesije nad bazom može se pristupiti već prethodno napravljenoj privremenoj tabeli i koristiti se njome bez potrebe za ponovnim pokretanjem skripta koji napravi privremenu tabelu. Sadržaj privremene tabele je privatan za svaku sesiju i moguće je pristupiti samo onim redovima podataka koji su ranije učitani u privremenu tabelu u okviru jedne konekcije. Orakl sintaksa za definisanje jedne privrmene tabele:

```
CREATE GLOBAL TEMPORARY TABLE Privremena_tabela
   (col1   NUMBER,
    col2     VARCHAR2(250),
    col3        VARCHAR2(3))
ON COMMIT PRESERVE ROWS;
```

DB2 baza od verzije 9.7 daje podršku za rad sa Orakl privremnim tabelama koji se očigledno dosta razlikuje od pristupa u radu sa DB2 privremenim tabelama(DGGT). Dodatne mogućnosti u radu sa Orakl privremenim tabelama su:

- Kada jedna sesija aplikacije definiše privremenu tabelu, konkurentne sesije nemaju potrebu za ponovnim definisanjem iste privremene tabele.
- Moguće je koristiti privremenu tabelu u procedurama, funkcijama, okidačima i pogledima u kombinaciji sa drugim privremenim ili trajnim tabelama baze.
- Ukoliko se u okviru jedne sesije koristi TRUNCATE naredba, onda su izbrisani samo oni podaci iz privremene tabele koji su karakteristični za tu sesiju bez uticaja na podatke koje su druge sesije sačuvale u istoj privremenoj tabeli.

- Nad jednom ili više kolona privremene tabele mogu biti definisani indeksi radi poboljšanja performansi prilikom čitanja podataka iz privremene tabele.

10. Zaključak

Migracija između RSUBP - ova Orakl i DB2, u bilo kom smeru nije trivijalna. Iako se radi o relacionim sistemima za upravljanje bazom podataka oni se mogu razlikovati u mnogim aspektima dizajna i arhitekture. Ove razlike mogu varirati i mogu biti različite prirode i složenosti.

Iskustvo ukazuje da dobro definisana metodologija može biti ključ uspeha jednog projekta migracije. Precizno definisane faze životnog ciklusa projekta migracije i tačno određeni poslovi koje je potrebno uraditi u njima garantuju uspešno izvršavanje i najsloženijih projekata. Detaljno planiranje poslova, vremenskih okvira i IT budžeta najvažnije su stavke u jednom uspešnom projektu migracije. Dobro razumevanje složenosti poslovne logike koja je pokrivena kroz implementirane objekte baze i aplikacije koje rade nad bazom može pomoći prilikom prepoznavanja svih izazova, složenosti ili napora koje je potrebno uložiti kada se migrira sa jednog relacionog sistema za upravljanje bazom podataka na drugi.

S druge strane, upotreba alata za migraciju koji su obezbeđeni od strane proizvođača relacionih sistema za upravljanje bazom podataka smanjuje troškove i rizik prilikom obavljanja poslova migracije. Jedna od glavnih procena koju je potrebno doneti u fazi analize i dizajna projekta migracije tiče se odabira skupa alata za migraciju koji zavisi od same prirode i složenosti poslova koje je potrebno obaviti. Ono što treba takođe uzeti u obzir prilikom odabira alata za migraciju jesu i razlike koje postoje među njima i koje se ogledaju u nivou automatizacije koju obezbeđuju i tačnosti migriranih SQL naredbi, definicija procedura, funkcija, okidača i aplikacija koje rade nad bazom.

Ono što je za jednu organizaciju od najvećeg značaja prilikom migracije baze na novi sistem jesu vremenski rokovi i troškovi prilikom obavljanja različitih poslova u okviru projekta migracije. Složeniji projekti koji uključuju migraciju više baza na ciljni sistem mogu trajati i više od godinu dana. U takvim slučajevima se najčešće primenjuje strategija razbijanja masivne migracije na nekoliko manjih projekata u kojima je cilj prvo migrirati manje baze pa posle toga vršiti migraciju složenijih baza.

U okviru ovoga rada predstavljen je pristup migraciji između RSUBP – ova Orakl i DB2 primenom alata za migraciju i dat je detaljan prikaz jednog životnog ciklusa projekta migracije. Cilj je bio da se informišu čitaoci o svim problemima koji se mogu javiti prilikom migracije između ova dva relaciona sistema i na koji način se ti problemi mogu rešiti.

11. Literatura

[1] Tom Laszewski, Prakash Nauduri. Migrating to the Cloud Oracle Client/Server Modernization. *Syngress, 2011*, *ISBN 1597496472.*

[2] Jason Williamson. Oracle Information Integration, Migration, and Consolidation. *Packt Publishing, 2011*, *ISBN 1849682208.*

[3] Whei-Jen Chen, Kenneth Chen, Patrick Dantressangle, Marina Greenstein, Sam Lightstone, Maksym Petrenko, Raanon Reutlinger, Serge Rielau, Steve Schormann, Maria N Schwenger. Oracle to DB2 Conversion Guide: Compatibility Made Easy. *IBM Redbooks 2009, SG247736.*

[4] Whei-Jen Chen, An Na Choi, Marina Greenstein, Scott J Martin, Fraser McArthur, Carlos Eduardo Abramo Pinto, Arthur V Sammartino, Nora Sokolof. Oracle to DB2 Conversion Guide for Linux, UNIX, and Windows. *IBM Redbooks 2007, SG247048.*

[5] Grant Allen. Beginning DB2 From Novice to Professional. *Apress 2008*, *ISBN 159059942X.*

[6] Oracle SQL Developer Migrations: Getting Started. [Na mreži]
http://www.oracle.com/technetwork/database/migration/omwb-getstarted-093461.html#prereq

[7] IBM Data Movement Tool, Move data from source databases to DB2 in an easy way. [Na mreži]
http://www.ibm.com/developerworks/data/library/techarticle/dm-0906datamovement/

[8] Leverage your Oracle 11g skills to learn DB2 9.7 for Linux, UNIX, and Windows. [Na mreži]
http://www.ibm.com/developerworks/data/library/techarticle/dm-0401gupta/

[9] DB2 10: Run Oracle applications on DB2 10 for Linux, UNIX, and Windows. [Na mreži]
http://www.ibm.com/developerworks/data/library/techarticle/dm-0907oracleappsondb2/

[10] Migrating from IBM DB2 LUW to Oracle. [Na mreži]
http://www.oracle.com/technetwork/database/migration/db2-084087.html