IZVANMREŽNA SINKRONIZACIJA SQL BAZA PODATAKA

Grubić, Roko

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Specijalistički diplomski stručni studij Informacijske tehnologije

ROKO GRUBIĆ

Z A V R Š N I R A D

IZVANMREŽNA SINKRONIZACIJA SQL BAZA PODATAKA

Split, srpanj 2021.

SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Specijalistički diplomski stručni studij Informacijske tehnologije

Predmet: Upravljanje poslužiteljima otvorenog kôda

Z A V R Š N I R A D

Kandidat: Roko Grubić

Naslov rada: Izvanmrežna sinkronizacija SQL baza podataka

Mentor: dipl.ing. Valentini Kožica, predavač

Split, srpanj 2021.

Sadržaj

SAŽETAK

Cilj ovog završnog rada je izrada skripti pomoću kojih se obavlja izvanmrežna sinkronizacija SQL (engl. *Structured Query Language*) baza podataka. U prvom dijelu rada je opisan koncept baza podataka i njihove potrebe u modernom svijetu. Opisane su tehnologije koje su potrebne za izradu i funkcioniranje ovakvog tipa sinkronizacije. Nakon toga je prikazan način izrade zadatka i implementacija korištenih tehnologija u gotovo rješenje, a na kraju je napisan zaključak o radu.

Za prikazani završni rad bio je potreban Microsoft SQL poslužitelj za baze podataka, jezik SQL koji služi za upravljanje bazama podataka te skriptni jezik PS (engl. *PowerShell*) unutar čijih skripti je implementirano rješenje. Virtualizator VMware Workstation Player korišten je za izradu okruženja sa dva poslužitelja na operacijskim sustavima Windows. Za pohranu podataka korišteni su CSV (engl. *Coma Separated Value*), XML (engl. *Extensible Markup Language*) i JSON (engl. *JavaScript Object Notation*) tipovi datoteka, a za njihov transfer korištena je usluga Dropbox.

Izvanmrežnom sinkronizacijom SQL baza podataka omogućeno je automatizirano ažuriranje podataka između poslužitelja koji nemaju međusobnu povezanost. Pod automatizacijom se podrazumijeva sinkronizacija preko rasporeda, odabirom programske opcije unutar određene aplikacije ili pozivanjem skripte izravno u naredbenoj liniji računala.

Ključne riječi: Microsoft SQL poslužitelj, skriptni jezik PS, CSV, XML, JSON

SUMMARY

Offline synchronization of SQL databases

Main goal of this final paper is creating scripts which are used for offline synchronization of SQL databases. Firstly, database concept and its necessity in modern world is explained. Also, technologies used for making this task are presented and described. After that, methods for implementation required technologies are given. In the end, there is conclusion about final paper.

For presented final paper Microsoft SQL Server was required, SQL language for database manipulation and PowerShell scripting language with which solution was implemented. VMware Workstation player is used for making environment with two SQL servers on different Windows operating systems. Storing data is resolved using CSV, XML and JSON files which are transfered using Dropbox service.

With offline synchronization of SQL databases it is possible to synchronize data between SQL servers that have no connection between them. Synchronizing data can be done with scheduled job, by clicking a button in application or calling scripts directly from computer command prompt.

Keywords: Microsoft SQL Server, PowerShell scripting language, CSV, XML, JSON

1. UVOD

Današnje društvo živi u dobu sve bržeg razvoja tehnologije i gotovo je nemoguće zamisliti moderni svijet bez računala. Kako su se računala razvijala, razvijala se i potreba za zapisom i pohranom podataka. Zbog takvih zahtjeva, danas postoje moderne baze podataka koje služe baš za tu svrhu, ali i puno više od toga. Prije postojanja baza podataka na računalima, sve informacije zapisivane su na papir što je predstavljalo nepraktično i relativno nesigurno rješenje.

Početkom 1960.-ih IBM (engl. *International Business Machines*) je razvio hijerarhijski model baza podataka koji predstavlja podatke kao zapise koji su spojeni sa poveznicama; svaki zapis ima svog roditelja koji započinje sa korijenom. 1969. godine znanstvenici na CODASYL-u (engl. *Conference on Data Systems Languages*) su predstavili mrežni model. Ovakav model dozvoljava veze između zapisa i dozvoljava više zapisa roditelja/djece te se stvaraju entiteti i atributi entiteta koji će biti spomenuti kasnije. Kod razvoja baza podataka treba spomenuti i datotečni model. Datotečni model baze podataka pohranjuje informacije u datoteke kojima upravlja aplikacija kojoj su te informacije potrebne. Struktura datoteka je definirana unutar programskog kôda.

Nakon ovih modela dolazi relacijski model (također predstavljen 1969.) koji pohranjuje podatke u tablice od kojih svaka ima atribute u koje se spremaju vrijednosti. Tip podataka koji se unosi je unaprijed određen i to donosi postojanost baze kao i lakši pronalazak potrebnih podataka. Također, relacijski model posjeduje veze između tablica koje pokazuju kako su tablice međusobno povezane. Većina relacijskih baza podataka koriste isti pristup kod dobivanja podataka: jezik SQL. SQL je standardizirani skriptni jezik za baze podataka koji na temelju naredbi i uvjeta filtrira i pronalazi potrebne rezultate. Također služi za kreiranje i postavljanje pravila vezanih uz samo bazu [1] . Relacijski model je model koji je najviše zaživio od prethodno spomenutih i predstavlja osnovu ovog završnog rada.

Baza podataka predstavlja organizirani skup informacija koji opisuju procese iz okoline, a podaci unutar baze predstavljaju dio realnog svijeta za koji je namjenjena. Moderne baze podataka imaju vlastite sustave za upravljanje koji predstavljaju alat za upravljanje podacima, njihovu zaštitu i olakšano korištenje informacija. Njihova osnova je entitet (tablica u bazi), odnosno skup objekata iz realnog svijeta koji imaju zajednička svojstva. Svojstva entiteta se nazivaju atributi (slogovi u tablici) [2].

Što se tiče relacija između tablica, razlikuju se tri tipa, a one su:

- Jedan-na-jedan – jedan zapis iz tablice A povezan je sa jednim zapisom u tablici B (i obrnuto)
- Jedan-na-više – jedan zapis iz tablice A može biti povezan sa više zapisa iz tablice B, a zapisi iz tablice B su povezani sa jednim zapisom u tablici A
- Više-na-više – više zapisa iz tablice A mogu biti povezani sa više zapisa iz tablice B (i obrnuto)

U sklopu objašnjenja relacijskih baza podataka, bitno je spomenuti termine primarnog i stranog ključa. Primarni ključ kratice PK (engl. *Primary Key*) predstavlja jedinstveni identifikator svakog redka tablice i ne mogu postojati dva redka u tablici sa istom vrijednošću primarnog ključa. Strani ključ kratice FK (engl. *Foreign Key*) predstavlja primarni ključ druge tablice koja je u relaciji sa tablicom u kojoj se strani ključ nalazi. Nakon izrade baze kreće se sa unosom podataka u skladu sa pravilima i namjenom baze.

U suvremenom svijetu postoji veliki broj aplikacija i korisnika koji preuzimaju podatke iz različitih baza iste namjene, a u svakom trenutku te baze moraju imati iste podatke kako bi isporučivali potrebne informacije. Javlja se problem što napraviti kako bi na bazama koje nisu međusobno povezane stalno imali iste podatke (bez obzira na unos, ažuriranje ili brisanje redaka iz tablica). Rješenje je predstavljeno u ovom završnom radu. Na dva poslužitelja koja se nalaze na različitim računalima napravljene su dvije vrste baza podataka (jedna sa istom, a druga sa različitom strukturom podataka) koje čuvaju podatke o pohađanju nastave na fakultetu.

Motivacija za rješavanje ovakvog problema proizlazi iz same svrhe baze podataka, a to je pohrana i pružanje spremljenih informacija koje računala, odnosno aplikacije dalje obrađuju i podatke prezentiraju krajnjim korisnicima.

Na Microsoft SQL poslužitelju kreirane su baze podataka u koje se unose podaci o pohađanju nastave. Uporabom skriptnog jezika PS sve SQL naredbe su ukomponirane unutar skripti i tako se baza kreira i popunjava. korištenjem funkcija unutar PS-a odabrani podaci se izvoze u CSV, XML i JSON datoteke te se preko servisa Dropbox šalju do drugog računala. Isti ovaj postupak se ponavlja i na drugom računalu i omogućena je izvanmrežna sinkronizacija baza podataka.

Izradom ovakve funkcije omogućeno je da se dvije baze koje se nalaze u potpuno različitim sustavima koji nemaju međusobnu povezanost, a moraju imati iste podatke, međusobno ažuriraju. Primjer je Hrvatski zavod za zdravstveno osiguranje i platforma Cijepise koji mogu skupljati podatke o cijepljenim osobama, a ne moraju biti međusobno povezani.

U slijedećem poglavlju dan je detaljniji pregled i opis korištenih tehnologija potrebnih za izradu završnog rada. Nakon toga slijedi opis izrade završnog rada uz potrebne priloge, a nakon toga je dan zaključak. Na samom kraju je navedena literatura koja je pomogla u izradi završnog rada.

1. KORIŠTENE TEHNOLOGIJE

2.1. Microsoft SQL Server

Prilikom konstruiranja baze podataka na računalu koja će kasnije ići u uporabu potrebno je instalirati poslužitelj. Poslužitelj baze podataka predstavlja računalo koje služi za pohranu i dobavljanje podataka, a podatke na udaljena mjesta šalje preko mreže. Svaki poslužitelj unutar sebe sadrži DBMS (engl. *Database Management System*) koji služi za komunikaciju sa bazom i samu bazu podataka u koju se podaci spremaju ili izvlače. Posebno je koristan organizacijama koje svakodnevno obrađuju veliku količinu informacija. Uobičajena arhitektura je tipa klijent – poslužitelj gdje korisnik šalje upite preko jezika SQL, a poslužitelj obrađuje zahtjeve i šalje rezultate natrag korisniku. Unutar većine poslužitelja postoje API-i (engl. *Application Programming Interface*) preko kojih se naredbe daju DBMS-u koji te naredbe dalje procesira [3]. Neki od najpoznatijih su:

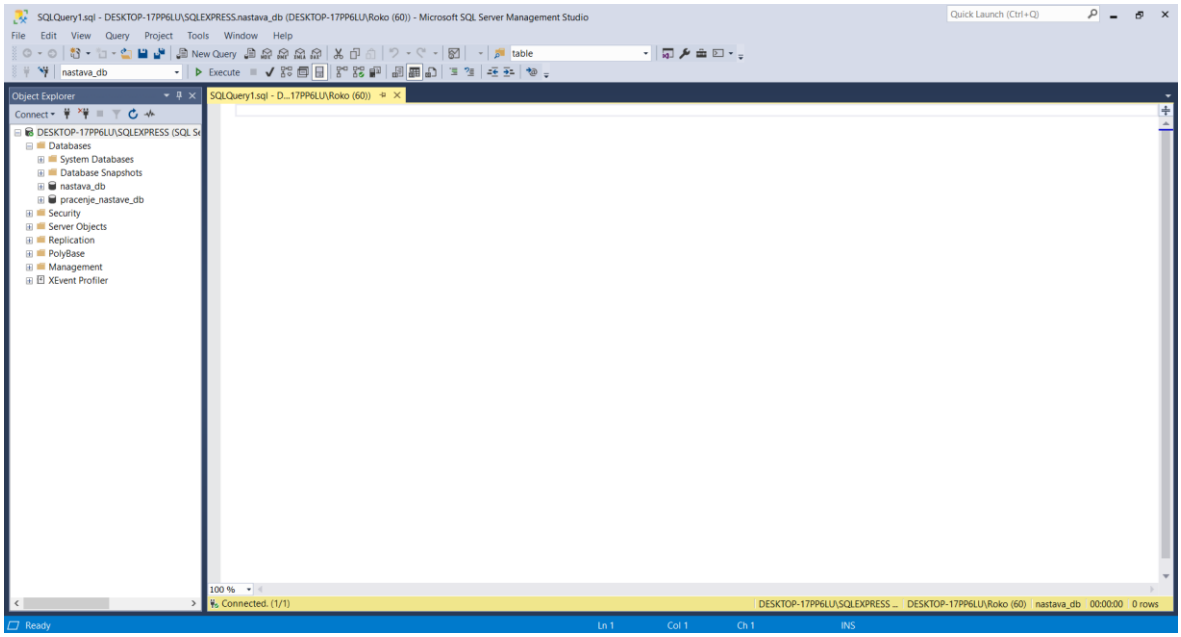
- Oracle
- PostgreSQL
- MySQL
- SQLite
- MSSQL

MSSQL (Microsoft SQL) poslužitelj je odabran za izradu ovog završnog rada, a razvijen je od strane Microsofta. Korištena je Express verzija 2019, a još postoje:

- Enterprise
- Standard
- Web
- Development

Upotrijebljena verzija je besplatna i ima svoja ograničenja. Ograničenja se odnose npr. na broj procesorskih jezgri (najviše 4 jezgre) korištenih kod obrade podataka kao i veličinu relacijske baze (do 10 GB). Uz dodatne alate koji dolaze sa poslužiteljem važno je spomenuti *SQL Server Management Studio* (skraćeno SSMS). SSMS je jedan od najbitnijih djelova zbog toga što je poveznica između korisnika i same baze, odnosno predstavlja API za upravljanje bazom podataka [4]. Alat je integrirano okruženje unutar kojeg se baze

podataka kreiraju, nadziru i administriraju, a pogodan je kako za razvojne inženjere tako i za administratore baza podataka bez obzira na razinu iskustva. Iako je priroda zadatka drukčija od klikanja na grafičkom sučelju, od izuzetne je koristi za pregled rezultata krajnjeg rješenja. SSMS sučelje može se vidjeti na slici 1.



Slika 1: SSMS sučelje

2.2. Skriptni jezik PS

PS je skriptni jezik koji se koristi za automatizaciju zadataka napravljenih unutar naredbene linije računala. Objavljen je 2006. godine i razvijen je pomoću programskog jezika C# i platforme net. Može se pokretati na operacijskim sustavima Windows, Linux i macOS. PS predstavlja nadogradnju osnovne naredbene linije (engl. *Command prompt*) koja komunicira sa ljuskom (engl. *shell*). Pod ljuskom se smatra okruženje unutar kojeg se računalom upravlja preko naredbi u naredbenoj liniji, a ne preko grafičkog sučelja. Svaka od njih ima svoju sintaksu pa se razlikuju u ovisnosti o operacijskom sustavu. Za razliku od većine ljuski koje samo prihvaćaju i vraćaju tekst, PS prihvaća i vraća net. objekte. Net. je razvojni okvir za izradu aplikacija, a sastoji se od više programskih jezika i također je dio Windows obitelji.

Kao ljuska, ovaj alat pruža slijedeće značajke:

- Povijest naredbi u naredbenoj liniji
- Predviđanje naredbi i završetak istih koristeći tipku Tab
- Podržava pseudonime naredbi i parametara
- Cjevovod za ulančavanje naredbi
- Sustav za pomoć unutar konzole

Kao skriptni jezik, PS se često koristi za automatizaciju upravljanja sustavima. Također, napravljen je za izradu, testiranje i postavljanje okruženja, a izrađen je na *.NET Common Language Runtime* koji predstavlja okruženje koje pokreće kôd i pruža usluge koje čine proces razvoja jednostavnijim [5]. Svi ulazi i izlazi su net. objekti, a pruža slijedeće značajke:

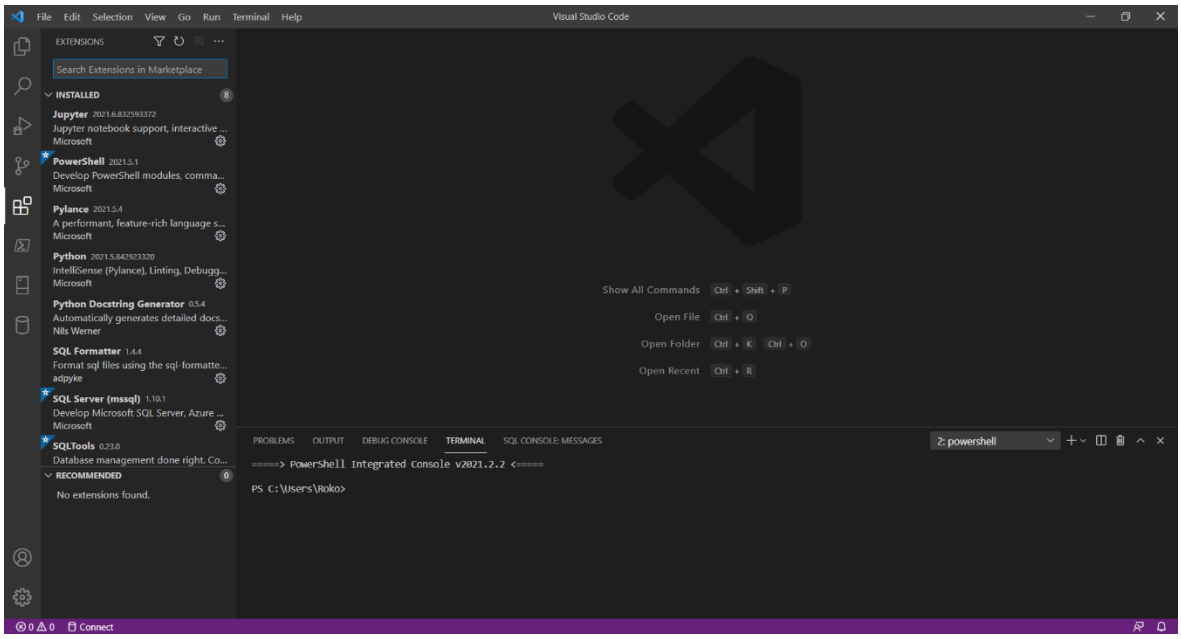
- Uporaba funkcija, klasa, skripti i modula
- Sustav formatiranja za jednostavniji prikaz rezultata
- Izrada dinamičkih tipova varijabli kroz proširiv sustav tipova
- Ugrađena podrška za tipove datoteka kao što su CSV, JSON i XML

Što se dodatnih modula tiče, svi potrebni moduli mogu se pronaći na *Powershell Gallery*, mjestu gdje se nalaze sva proširenja za ovaj skriptni jezik. PS je u završnom radu korišten za enkapsulaciju SQL naredbi kao i za upravljanje podacima dobivenih iz datoteka potrebnih za sinkronizaciju. Tipičan izgled kôda koji koristi funkciju za uvoz datoteke i onda svaki net. objekt preko cjevovoda ispisuje u konzolu prikazan je u ispisu 1.

```
$csv_student = Import-Csv -Path
„E:\Zavrsni_ispit\Datoteke_triggers\Csv_datoteke\Promjene_student.csv“

$csv_student | ForEach-Object {

      Write-Host $_

}
```

Ispis 1: Primjer kôda u PS-u

Sav kôd za završni rad pisan je u Visual Studio Code-u, lako proširivom IDE-u (engl. *Integrated Development Environment*) u kojem je izrazito jednostavno dodati proširenja za gotovo sve programske jezike, uključujući i PS. Podržava i traženje pogrešaka u kôdu kao i verzioniranje rješenja sa npr. *GitHubom.* Prikaz početnog zaslona Visual Studio Codea je prikazan na slici 2.

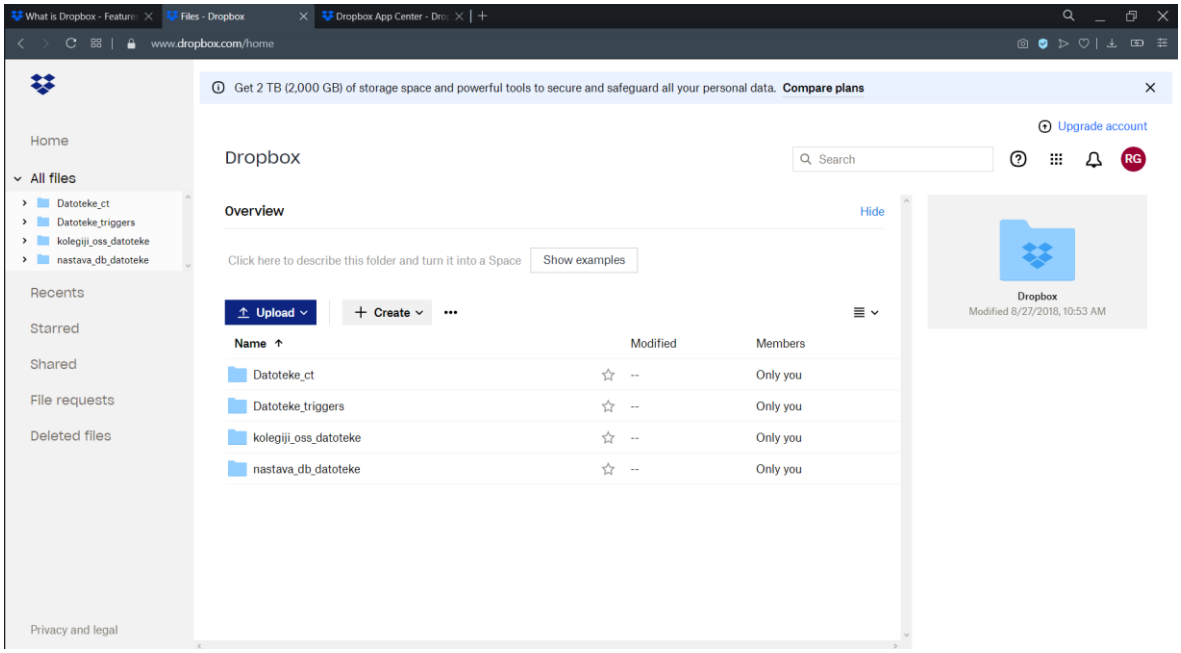


Slika 2: Izgled Visual Studio Code okruženja

2.3. Dropbox

Dropbox je usluga za pohranu datoteka preko mreže koja je nastala 2007. godine, a sjedište joj je u San Franciscu, Kalifornija, SAD. Izrađena je u programskim jezicima Python, Go, CoffeScript i Rust. Osim pohrane datoteka, pruža usluge sinkronizacije kao i usluge klijentskih aplikacija. Unutar Dropboxa moguće je povezivati se sa drugim aplikacijama koje služe za komunikaciju i kolaboraciju kao što su Zoom, Slack i Trello [6]. Od ostalih aplikacija sa kojima se može povezivati tu su Microfost Office paket, AutoCAD, itd.. Izrazito je koristan za tvrtke i organizacije koje žele imati svoje podatke na jednom mjestu i dodati sigurnost s obzirom na to da se podaci ne nalaze na lokalnim računalima nego su na mreži, a Dropbox jamči da će podaci biti sigurni. Uz povezivanje sa vanjskim aplikacijama, moguće je kreirati i osobne aplikacije preko kojih se može upravljati sa dokumentima na računu. Također, treba spomenuti kako Dropbox ima svoj API koji služi za upravljanje Dropboxom kao i za postavljanje i preuzimanje mapa i datoteka. Za tu svrhu je

i korišten, odnosno preko skriptnog jezika PS je napravljena funkcija za postavljanje i funkcija za preuzimanje datoteka koje su kasnije korištene za ažuriranje tablica u bazama. Izgled korisničkog sučelja Dropbox računa može se vidjeti na slici 3.



Slika 3: Dropbox sučelje

2.4. Tipovi datoteka

2.4.1. CSV datoteka

Najjednostavniji tip datoteke u koju se podaci izvoze naziva se CSV datoteka. Kao što joj skraćenica kaže, CSV sprema podatke u red gdje je svaka nova vrijednost u redu odvojena interpunkcijskim znakom, najčešće zarezom, a svaki novi zapis ide u slijedeći red. Često je korištena za pohranu podataka i upravljačke zadatke nad tim podacima. Vrlina ovog tipa datoteke leži u njenoj jednostavnosti i primarno je korištena za pohranu tabličnih informacija koje se kasnije mogu prebaciti u redke i stupce. Koristi se u poslovnim domenama kako bi organizacije mogle izvesti svoje podatke u druge baze podataka lako i jednostavno. Jednako tako je korištena u znanstvenim i inženjerskim područjima kao i u proizvodnoj te zdravstvenoj industriji. Prvi službeni spomen CSV datoteke dolazi 1983. godine, a službeni standard za format naziva se RFC 4180. Primjer formata datoteke prikazan je u ispisu 2.

```
Ime, Prezime, Datum_rodenja, OIB

Ana, Anić, 01.01.2000., 12345678910

Ivo, Ivić, 02.02.1999., 10987654321
```

Ispis 2: CSV format

Prednosti ovog tipa datoteke su mnogostruke, a neke od njih su:

- Jednostavno kreiranje CSV datoteke koristeći bilo koji tekstualni uređivač
- Jednostavno čitanje jer podaci nisu kodirani u npr. binarni oblik prije spremanja
- Može biti otvorena koristeći bilo koji tekstualni uređivač
- Jednostavna za obradu i upravljanje podacima
- Rezultira malom veličinom datoteka s obzirom na jednostavnost formata
- Kompaktan oblik; nema potrebe za oznakama kao kod npr. XML ili JSON datoteka

Iako je ovdje nabrojano dosta prednosti, postoje i određene mane kao što je nedovoljno dobra podrška za složene podatke [7]. Još jedan nedostatak predstavlja spremanje podataka kao tekst bez obzira na njihov stvarni tip pri izvozu. Stoga je potrebno naći rješenje za pretvorbu podataka u prave tipove, kako za CSV, tako i za ostale vrste datoteka.

2.4.2. XML datoteka

Slijedeća vrsta datoteke korištena u zadatku je XML. XML je jezik za označavanje koji se počeo koristiti 90-ih godina 20. stoljeća i postao je dosta popularan za prijenos podataka preko interneta. Služi za spremanje podataka prema unaprijed određenoj shemi kako bi mogao biti lako iščitavan i dijeljen između aplikacija.

Format se sastoji od dva dijela: čistog teksta i oznaka. Čisti tekst predstavlja stvarnu vrijednost nekog atributa dok oznaka predstavlja tip podatka (ime, prezime i sl.). Svaka XML oznaka se naziva element, a elementi su poredani hijerarhijski. Prva oznaka je korijen datoteke i sadrži sve ostale elemente koji se nazivaju djeca. Često su djeca uvučena kako bi se olakšalo čitanje, ali to ne utječe na računalnu obradu XML datoteke [8].

Format XML datoteke može se vidjeti u ispisu 3.

```
<?xml version="1.0" encoding="utf-8">
<Studenti>
	<Student>
		<Ime>Ana</Ime>
		<Prezime>Anić</Prezime>
		<Datum_rodenja>01.01.2000.</Datum_rodenja>
		<OIB>12345678910</OIB>
	</Student>
	<Student>
		<Ime>Ivo</Ime>
		<Prezime>Ivić</Prezime>
		<Datum_rodenja>02.02.1999.</Datum_rodenja>
		<OIB>10987654321</OIB>
	</Student>
</Studenti>
```

Ispis 3: XML format

Koristi se kod prijenosa digitalnih informacija između internetskih poslužitelja, u računalnim aplikacijama za dobavu podataka, za prikaz podataka na internetskim stranicama pa i za prijenos podataka iz jedne u drugu bazu. Neke od prednosti XML tipa datoteke su:

- Neovisan je o platformi i programskom jeziku koji ga koristi
- Ima podršku za Unicode, međunarodni standard za kodiranje svih znakova pa time omogućava XML-u da prenosi bilo kakvu vrstu informacija
- Podaci koji se prenose preko ovog tipa datoteka mogu biti mjenjani u bilo kojem trenutku bez utjecaja na prezentaciju podataka

Nadalje, što se tiče nedostataka može se spomenuti manja čitljivost od npr. JSON formata, ne podržava niz kao vrijednost podatka i veličina datoteka je dosta velika s obzirom na mogućnosti oznaka i vrijednosti te ovisi o tome tko izrađuje XML datoteku [9].

2.4.3. JSON datoteka

Trenutno najpopularnija datoteka za prijenos podataka je JSON i za to ima valjane razloge. JSON je razvijen 2001. godine kako bi riješio problem prijenosa JavaScript podataka preko HTML jezika (engl. *HyperText Markup Language*). Svoj konačan oblik dobio je 2002. godine i od tada ima isti format. Format je univerzalan bez obzira na platformu i programski jezik koji obrađuje datoteku, a može se vidjeti u ispisu 4.

```
[
      {
         „Ime“ : „Ana“
         „Prezime“ : „Anić“
         „Datum_rodenja“ : „01.01.2000.“
         „OIB“ : 12345678910
      },
      {
         „Ime“ : „Ivo“
         „Prezime“ : „Ivić“
         „Datum_rodenja“ : „02.02.1999.“
         „OIB“ : 10987654321
      }
]
```

Ispis 4: JSON format

U usporedbi sa XML-om puno je jednostavniji za pisanje. Temelji se na JavaScript programskom jeziku čiji se objekti lako kreiraju. JSON koristi navodne znakove za ključeve (koji predstavljaju naziv vrijednosti) i tekstualne vrijednosti, a još podržava tipove podataka null, boolean, broj, objekt i niz.

Za notaciju objekta potrebne su valovite zagrade, a za niz uglate [10]. Uzimajući u obzir univerzalnost, JSON se koristi za mnogo stvari, a najviše za generiranje objekata sastavljenih od korisničkih podataka (forma na internetskoj stranici). Također, koristi se za prijenos podataka između sustava (sinkronizacija baza), pristupne podatke za aplikacije i za pojednostavljivanje kompleksnih modela podataka.

Neke od prednosti JSON-a su:

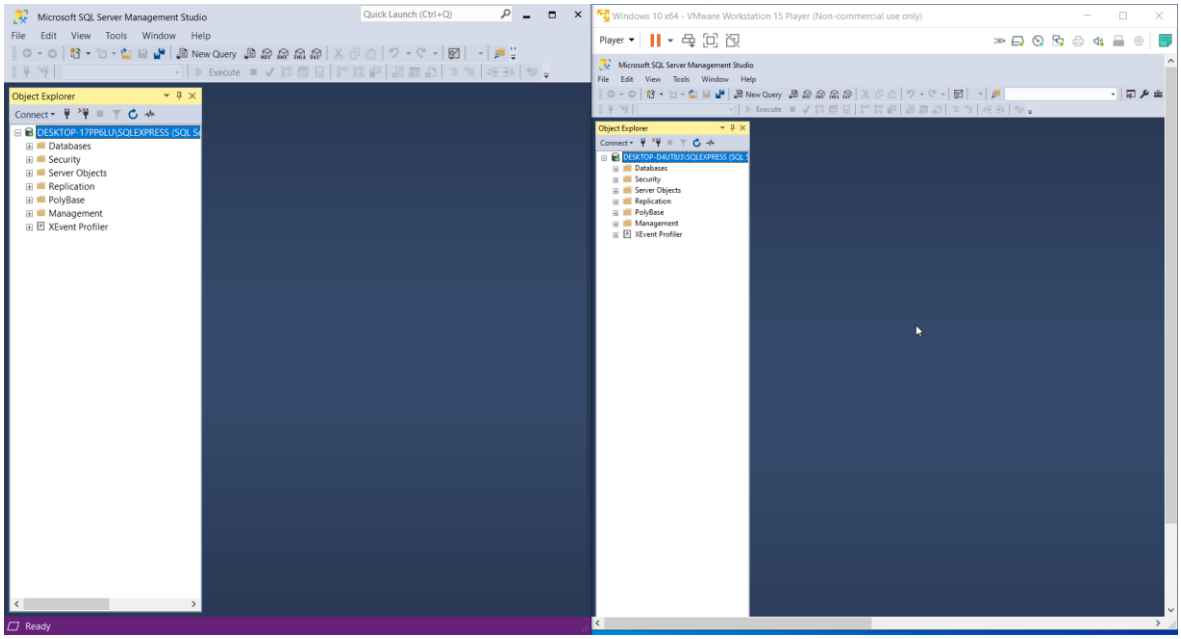
- Sintaksa je jednostavna za korištenje i ne zauzima puno prostora pa je samim time JSON brži u usporedbi sa XML-om
- Kompatibilan je sa više platformi
- Jednostavno je izvlačiti podatke iz JSON datoteke
- Moguće je pohranjivati podatke u nizove pa je moguće prenositi čak i audio i video zapise

Što se tiče mana, JSON nema podršku za upravljanje greškama kod poziva pa istu nije moguće vidjeti. Treba biti oprezan sa JSON datotekama budući da nisu previše sigurne i mogu biti izložene napadima ukoliko se otvaraju preko neprovjerenih preglednika ili internetskih stranica [11].

3. POSTUPAK IZRADE ZAVRŠNOG RADA

3.1. Postavljanje okruženja

Prije početka izrade modela baze podataka i skripti koje su potrebne, postavljeno je okruženje unutar kojeg se zadatak izrađivao. Na računalu domaćinu (operativni sustav Windows 10 Education) instaliran je SQL Server 2019 Express sa SSMS-om verzije 18.8. Nakon instalacije za pristup poslužitelju postavljena je Windows autentifikacija i poslužitelj je bio spreman za rad. Isto tako, na računalu domaćinu je instaliran Visual Studio Code, prethodno spomenuti IDE preko kojeg su pisane kako PS, tako i SQL skripte. Nakon provjere funkcionalnosti, dodan je virtualizator VMware Workstation Player. VMware služi za virtualizaciju drugog operativnog sustava na kojem se nalazi „udaljeni“ poslužitelj. Sa službene Windows stranice skinuta je Windows 10 Education verzija operativnog sustava i dodana je u VMware. Na drugom operativnom sustavu također je instaliran SQL Server 2019 Express sa identičnim postavkama kao i na računalu domaćinu, a za prilagodbu kôda korišten je *PowerShell IDE*, okruženje koje dolazi u paketu sa Windowsom i preporučeno je za izradu PS skripti. Nakon postavljanja, poslužitelji koji se nalaze na dva različita računala i nisu međusobno spojeni instalirana su i spremna za rad. Snimka zaslona može se vidjeti na slici 4.

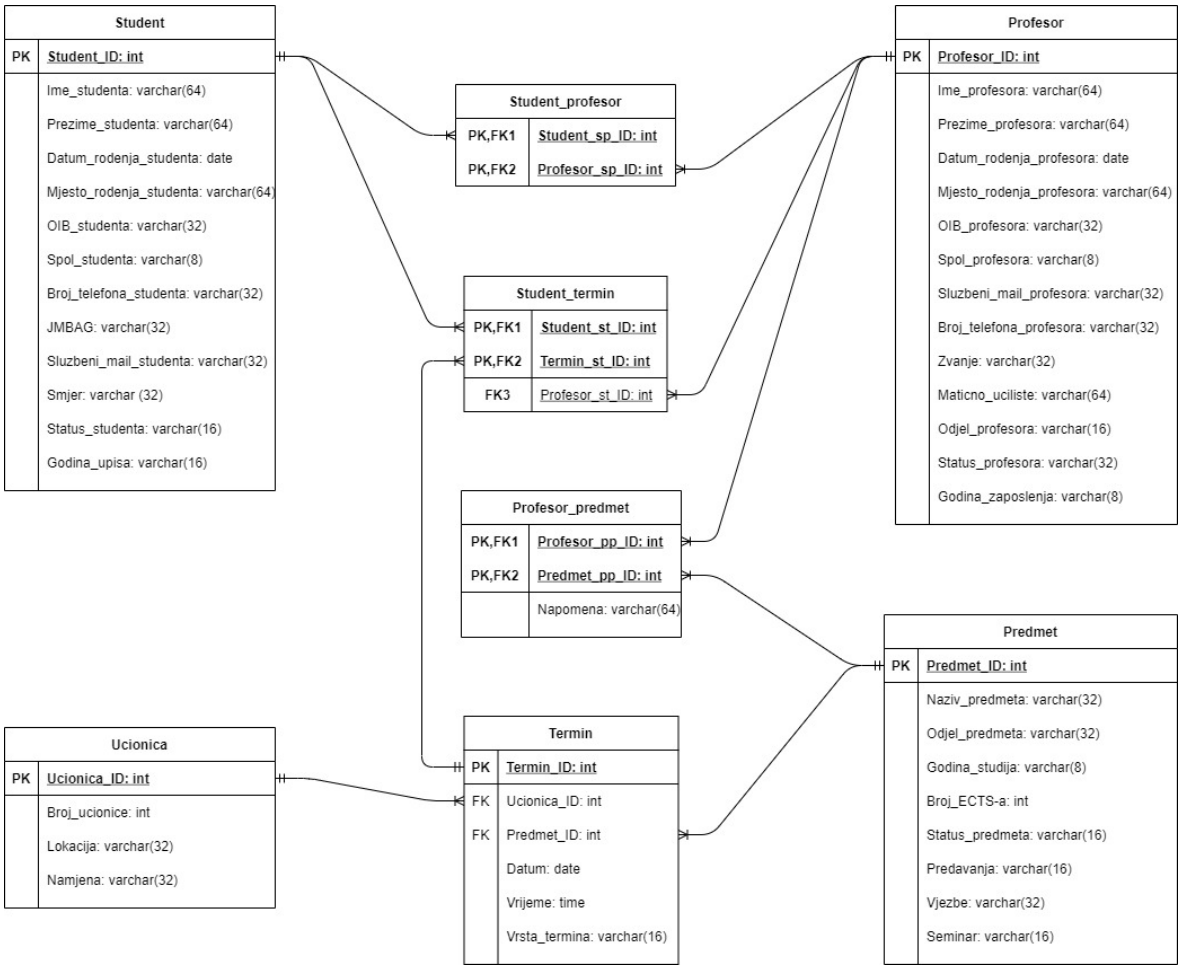


Slika 4: Postavljeno okruženje

3.2. Kreiranje baza podataka

3.2.1. Identične baze podataka

Kako je spomenuto u uvodu, u zadatku se radilo sa bazom podataka za praćenje nastave na fakultetu. Dijagram entiteti-veze prikazan je na slici 5.



Slika 5: Model baze podataka pracenje_nastave_db

Analizom potrebnih entiteta i atributa došlo se do 8 tablica koje su potrebne kako bi sustav praćenja nastave funkcionirao. Baza podataka naziva se *pracenje_nastave_db*, a tablice su Student, Profesor, Predmet, Ucionica, Termin, Student_profesor, Profesor_predmet i Student_termin. Tablice su napravljene u sklopu oba poslužitelja koji se nalaze na različitim računalima. Međusobne relacije su tipa jedan-na-više, a termini PK i FK predstavljaju pripadajuće primarne i strane ključeve tablica u relaciji. Tablice Student_profesor, Profesor_predmet i Student_termin predstavljaju posebno pravilo kod

relacijskih baza. Pravilo glasi da tablice koje imaju vezu tipa više-na-više (u prikazanom slučaju to su Student i Profesor, Profesor i Predmet te Student i Termin) pri izradi baze stvaraju međurelacijsku tablicu tipa jedan-na-više. Primarni ključevi tablica ujedno su primarni i strani ključevi nove tablice (uz moguće dodatne atribute koje takva veza stvara).

Model podataka prikazan je u nastavku, a entitet Student prikazan u tablici 1.

Tablica 1: Tablica Student

Student			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Student_ID	int	/	(1,1)
Ime_studenta	varchar	64	(1,1)
Prezime_studenta	varchar	64	(1,1)
Datum_rodenja_studenta	date	/	(1,1)
Mjesto_rodenja_studenta	varchar	64	(0,1)
OIB_studenta	varchar	32	(1,1)
Spol_studenta	varchar	8	(1,1)
Broj_telefona_studenta	varchar	32	(0,1)
JMBAG	varchar	32	(1,1)
Sluzbeni_mail_studenta	varchar	32	(0,1)
Smjer	varchar	32	(1,1)
Status_studenta	varchar	16	(1,1)
Godina_upisa	varchar	16	(1,1)

U tablicama je vidljiv stupac Kardinalitet za svaki atribut, a po definiciji kardinalitet atributa predstavlja broj koji govori koliko vrijednosti pojedini atribut daje za opis jednog elementa entiteta. Postoji donja i gornja granica kardinaliteta i kreću se u rasponu od 0, 1 i n, odnosno više vrijednosti.

Entitet Profesor je prikazan u tablici 2.

Tablica 2: Tablica Profesor

Profesor			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Profesor_ID	int	/	(1,1)
Ime_profesora	varchar	64	(1,1)
Prezime_profesora	varchar	64	(1,1)
Datum_rodenja_profesora	date	/	(1,1)
Mjesto_rodenja_profesora	varchar	64	(0,1)
OIB_profesora	varchar	32	(1,1)
Spol_profesora	varchar	8	(1,1)
Sluzbeni_mail_profesora	varchar	32	(0,1)
Broj_telefona_profesora	varchar	32	(0,1)
Zvanje	varchar	32	(1,1)
Maticno_uciliste	varchar	64	(0,1)
Odjel_profesora	varchar	16	(1,1)
Status_profesora	varchar	32	(1,1)
Godina_zaposlenja	varchar	8	(0,1)

Entitet Predmet prikazan je u tablici 3.

Tablica 3: Tablica Predmet

Predmet			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Predmet_ID	int	/	(1,1)
Naziv_predmeta	varchar	32	(1,1)
Odjel_predmeta	varchar	32	(1,1)
Godina_studija	varchar	8	(1,1)
Broj_ECTSa	int	/	(1,1)
Status_predmeta	varchar	16	(1,1)
Predavanja	varchar	16	(0,1)
Vjezbe	varchar	32	(0,1)
Seminar	varchar	16	(0,1)

Entitet Ucionica prikazan je u tablici 4.

Tablica 4: Tablica Ucionica

Ucionica			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Ucionica_ID	int	/	(1,1)
Broj_ucionica	int	/	(1,1)
Lokacija	varchar	32	(1,1)
Namjena	varchar	32	(0,1)

Entitet Termin prikazan je u tablici 5.

Tablica 5: Termin

Termin			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Termin_ID	int	/	(1,1)
(FK) Ucionica_termin_ID	int	/	(1,1)
(FK) Predmet_termin_ID	int	/	(1,1)
Datum	date	/	(0,1)
Vrijeme	time	/	(0,1)
Vrsta_termina	varchar	16	(0,1)

Entitet Student_profesor prikazan je u tablici 6.

Tablica 6: Student_profesor

Student_profesor			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) (FK) Student_sp_ID	int	/	(1,1)
(PK) (FK) Profesor_sp_ID	int	/	(1,1)

Entitet Profesor_predmet prikazan je u tablici 7.

Tablica 7: Profesor_predmet

Profesor_predmet			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) (FK) Profesor_pp_ID	int	/	(1,1)
(PK) (FK) Predmet_pp_ID	int	/	(1,1)
Napomena	varchar	64	(0,1)

Entitet Student_termin prikazan je u tablici 8.

Tablica 8: Tablica Student_termin

Student_termin			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) (FK) Student_st_ID	int	/	(1,1)
(PK) (FK) Termin_st_ID	int	/	(1,1)
(FK) Profesor_st_ID	int	/	(1,1)

U sklopu jednog od rješenja zadatka unutar baze su dodane takozvane *export* tablice, odnosno tablice koje imaju ista polja (uz atribut operacije i primarnog ključa) kao i tablice koje se nalaze u relaciji. *Export* tablice služe za spremanje redaka koji su izmjenjeni i nisu međusobno povezane, a samo rješenje će biti detaljnije objašnjeno u nastavku rada.

Kako se u podlozi nalazi SQL poslužitelj, potrebno je napisati kôd koji izrađuje bazu podataka i tablice. Kôd za izradu tablica spremljen je pod nazivom *napravi_tablice.sql* i prikazan je u ispisu 5.

```
...
CREATE TABLE Termin(
    Termin_ID INT NOT NULL,
    Ucionica_termin_ID INT NOT NULL,
    Predmet_termin_ID INT NOT NULL,
    Datum DATE,
    Vrijeme TIME,
    Vrsta_termina VARCHAR(16),
    CONSTRAINT Termin_PK PRIMARY KEY(Termin_ID),
    CONSTRAINT Termin_ucionica_FK FOREIGN KEY(Ucionica_termin_ID)
REFERENCES Ucionica(Ucionica_ID)
    ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT Termin_predmet_FK FOREIGN KEY(Predmet_termin_ID)
REFERENCES Predmet(Predmet_ID)
    ON DELETE CASCADE ON UPDATE CASCADE
);
...
CREATE TABLE Termin_export(
    Promjena_termin_ID INT IDENTITY(1,1) NOT NULL,
    Tip_promjene_t VARCHAR(16) NOT NULL,
    Termin_export_ID INT NOT NULL,
    Ucionica_termin_export_ID INT NOT NULL,
    Predmet_termin_export_ID INT NOT NULL,
    Datum_export DATE,
    Vrijeme_export TIME,
    Vrsta_termina_export VARCHAR(16),
    CONSTRAINT Termin_export_PK PRIMARY KEY(Promjena_termin_ID)
);
...
```

Ispis 5: Dio skripte napravi_tablice.sql

Unutar kôda vide se nazivi, veličine i tip polja te ključna naredba `CREATE TABLE` sa kojom se kreiraju tablice. Ključnom riječju `CONSTRAINT` dodavaju se pravila na tablicu i/ili bazu te ih je na taj način lakše mjenjati i brisati. `PRIMARY KEY` označava primarni ključ tablice, a `FOREIGN KEY` strani ključ. `UNIQUE` označava polje koje mora biti jedinstveno i ne smije se pojavljivati u niti jednom drugom redu.

Da bi postojali podaci koje treba sinkronizirati napisana je SQL skripta za punjenje tablica naziva *napuni_tablice.sql*, a vidi se u ispisu 6.

```
...
INSERT INTO Termin(Termin_ID, Ucionica_termin_ID, Predmet_termin_ID, Datum, Vrijeme, Vrsta_termina)
VALUES (1, 1, 3, '2020-10-02', '16:15', 'Predavanja'),
(2, 3, 1, '2020-10-02', '19:15', 'LV'),
(3, 2, 2, '2020-10-03', '09:15', 'Predavanja');
...
```

Ispis 6: Dio skripte napuni_tablice.sql

Unutar skripte ključna je naredba `INSERT INTO` nakon koje ide naziv tablice u koju se podaci umeću i njena polja. Kod ubacivanja podataka potrebno je paziti na ispravnost tipa podatka koji se unosi, inače se javlja greška.

PS se koristi za enkapsulaciju SQL naredbi i komunikaciju sa poslužiteljem. Kako bi se skripta uspješno povezala sa poslužiteljem i izvršila naredbe sa *PowerShell Gallerya* preuzet je modul *sqlserver* jednostavnom naredbom `Install-Module sqlserver`. Kôd preko kojeg se kreira baza podataka i sve njene tablice spremljen je u skriptu *inicijaliziraj_bazu.ps1*, a može se vidjeti u ispisu 7.

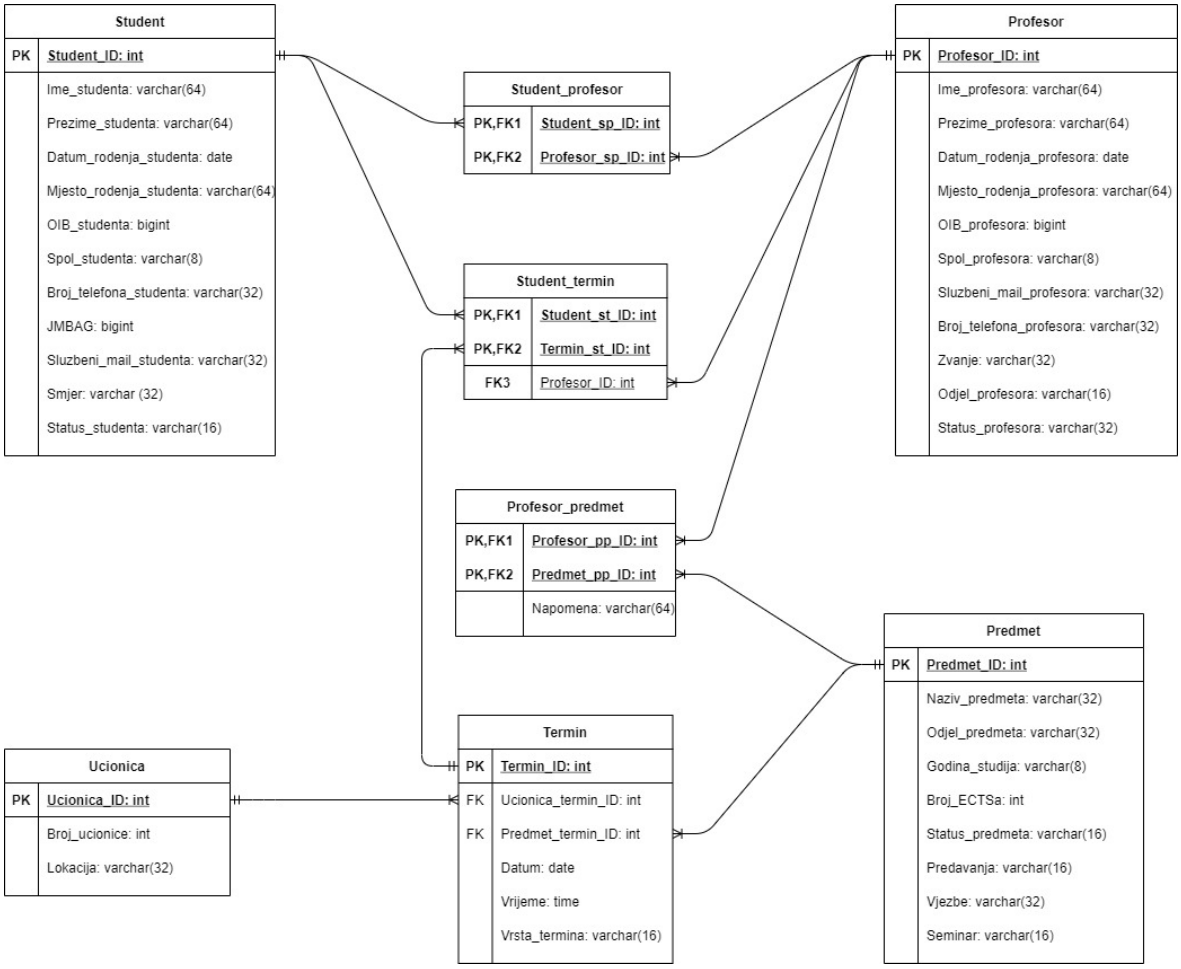
```
$server = "DESKTOP-17PP6LU\SQLEXPRESS"

$db = "pracenje_nastave_db"

Invoke-Sqlcmd -ServerInstance $server -Query "CREATE DATABASE $db ON (NAME
= pracenje_nastave_db_data,

FILENAME                                =                                'E:\SQL
Express\MSSQL15.SQLEXPRESS\MSSQL\DATA\pracenje_nastave_db_data.mdf', SIZE
= 8MB, MAXSIZE = 6GB, FILEGROWTH = 5MB)

LOG ON (NAME = pracenje_nastave_db_log,

FILENAME                                =                                'E:\SQL
Express\MSSQL15.SQLEXPRESS\MSSQL\DATA\pracenje_nastave_db_log.ldf',  SIZE
= 4MB, MAXSIZE = 3GB, FILEGROWTH = 5MB);"

Invoke-Sqlcmd -ServerInstance $server -Database $db -Query "USE $db; ALTER
DATABASE $db SET RECOVERY FULL;"

Invoke-Sqlcmd   -ServerInstance   $server   -Database   $db   -InputFile
"E:\Zavrsni_ispit\Skripte\SQL\napravi_tablice.sql"

Invoke-Sqlcmd   -ServerInstance   $server   -Database   $db   -InputFile
"E:\Zavrsni_ispit\Skripte\SQL\napuni_tablice.sql"

Invoke-Sqlcmd   -ServerInstance   $server   -Database   $db   -InputFile
"E:\Zavrsni_ispit\Skripte\SQL\svi_okidaci.sql"

Invoke-Sqlcmd   -ServerInstance   $server   -Database   $db   -InputFile
"E:\Zavrsni_ispit\Skripte\SQL\enable_tracking.sql"
```

Ispis 7: Skripta inicijaliziraj_bazu.ps1

Varijabla `$server` predstavlja naziv poslužitelja, a varijabla `$db` naziv baze podataka. `CREATE DATABASE` je ključna naredba za izradu baze podataka, a nakon nje slijede parametri za postavke mjesta izrade baze kao i njene veličine. Također, dodana je i naredba za postavljanje sigurnosne kopije (engl. *backup*) u slučaju pogreške u bazi. `Invoke-Sqlcmd` je naziv funkcije *sqlserver* modula koja se uz pomoć parametara `-ServerInstance` za naziv poslužitelja i `-Database` za naziv baze spaja na poslužitelj i izvršava SQL naredbe u PS-u. `-InputFile` je parametar korišten kako bi se naredba uputila na mjesto pohrane prethodno spomenutih skripti koji služe za kreiranje i popunjavanje tablica. Na taj način, sve potrebno se nalazi unutar skripte koju je potrebno pokrenuti jednom te nije potrebno pokretati svaki kôd zasebno. Izvršavanjem ovog kôda napravljena je baza podataka, njene tablice i umetnuti su podaci unutar tablica. Isti ovaj postupak napravljen je i na drugom poslužitelju uz izmjenu naziva poslužitelja i putanje do SQL skripti.

3.2.2. Različite baze podataka

Drugi dio zadatka donosi sinkronizaciju baza podataka sa različitom strukturom i tipom podataka koja se također odnosila na praćenje nastave na fakultetu. Dijagram entiteti-veze može se vidjeti na slici 6.



Slika 6: Model baze podataka nastava_db

Na prvom poslužitelju napravljena je baza podataka naziva *nastava_db* koja se sastoji od 8 tablica (Student, Profesor, Predmet, Ucionica, Termin, Student_profesor, Profesor_predmet i Student_termin). Napravljena baza predstavlja malo izmjenjenu bazu *pracenje_nastave_db* iz prijašnjeg primjera. Odnosi između tablica su isti kao i u primjeru gdje su na dva različita poslužitelja iste baze podataka, a model podataka prikazan je u nastavku. Bitno je napomenuti kako unutar ovog dijela zadatka nema *export* tablica već postoji samo jedna tablica unutar koje se stavljaju svi podaci naziva Podaci_transfer.

Entitet Student prikazan je u tablici 9.

Tablica 9: Tablica Student

Student			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Student_ID	int	/	(1,1)
Ime_studenta	varchar	64	(1,1)
Prezime_studenta	varchar	64	(1,1)
Datum_rodenja_studenta	date	/	(1,1)
Mjesto_rodenja_studenta	varchar	64	(0,1)
OIB_studenta	bigint	/	(1,1)
Spol_studenta	varchar	8	(1,1)
Broj_telefona_studenta	varchar	32	(0,1)
JMBAG	bigint	/	(1,1)
Sluzbeni_mail_studenta	varchar	32	(0,1)
Smjer	varchar	32	(1,1)
Status_studenta	varchar	16	(1,1)

Entitet Profesor prikazan je u tablici 10.

Tablica 10: Tablica Profesor

Profesor			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Profesor_ID	int	/	(1,1)
Ime_profesora	varchar	64	(1,1)
Prezime_profesora	varchar	64	(1,1)
Datum_rodenja_profesora	date	/	(1,1)
Mjesto_rodenja_profesora	varchar	64	(0,1)
OIB_profesora	bigint	/	(1,1)
Spol_profesora	varchar	8	(1,1)
Sluzbeni_mail_profesora	varchar	32	(0,1)
Broj_telefona_profesora	varchar	32	(0,1)
Zvanje	varchar	32	(1,1)
Odjel_profesora	varchar	16	(1,1)
Status_profesora	varchar	32	(1,1)

Entitet Predmet prikazan je u tablici 11.

Tablica 11: Tablica Predmet

Predmet			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Predmet_ID	int	/	(1,1)
Naziv_predmeta	varchar	32	(1,1)
Odjel_predmeta	varchar	32	(1,1)
Godina_studija	varchar	8	(1,1)
Broj_ECTSa	int	/	(1,1)
Status_predmeta	varchar	16	(1,1)
Predavanja	varchar	16	(0,1)
Vjezbe	varchar	32	(0,1)
Seminar	varchar	16	(0,1)

Entitet Ucionica prikazan je u tablici 12.

Tablica 12: Tablica Ucionica

Ucionica			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Ucionica_ID	int	/	(1,1)
Broj_ucionica	int	/	(1,1)
Lokacija	varchar	32	(1,1)

Entitet Termin prikazan je u tablici 13.

Tablica 13: Tablica Termin

Termin			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Termin_ID	int	/	(1,1)
(FK) Ucionica_termin_ID	int	/	(1,1)
(FK) Predmet_termin_ID	int	/	(1,1)
Datum	date	/	(0,1)
Vrijeme	time	/	(0,1)
Vrsta_termina	varchar	16	(0,1)

Entitet Student_profesor prikazan je u tablici 14.

Tablica 14: Student_profesor

Student_profesor			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) (FK) Student_sp_ID	int	/	(1,1)
(PK) (FK) Profesor_sp_ID	int	/	(1,1)

Entitet Profesor_predmet prikazan je u tablici 15.

Tablica 15: Tablica Profesor_predmet

Profesor_predmet			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) (FK) Profesor_pp_ID	int	/	(1,1)
(PK) (FK) Predmet_pp_ID	int	/	(1,1)
Napomena	varchar	64	(0,1)

Entitet Student_termin prikazan je u tablici 16.

Tablica 16: Tablica Student_termin

Student_termin			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) (FK) Student_st_ID	int	/	(1,1)
(PK) (FK) Termin_st_ID	int	/	(1,1)
(FK) Profesor_st_ID	int	/	(1,1)

SQL skripta *napravi_tablice.sql* vidljiva je u ispisu 8.

```
CREATE TABLE Student(
    Student_ID INT NOT NULL,
    Ime_studenta VARCHAR(64) NOT NULL,
    Prezime_studenta VARCHAR(64) NOT NULL,
    Datum_rodenja_studenta DATE NOT NULL,
    Mjesto_rodenja_studenta VARCHAR(64),
    OIB_studenta BIGINT NOT NULL,
    Spol_studenta VARCHAR(8) NOT NULL,
    Broj_telefona_studenta VARCHAR(32),
    JMBAG BIGINT NOT NULL,
    Sluzbeni_mail_studenta VARCHAR(32),
    Smjer VARCHAR(32) NOT NULL,
    Status_studenta VARCHAR(16) NOT NULL,
    CONSTRAINT Student_PK PRIMARY KEY(Student_ID),
    CONSTRAINT  Student_uq  UNIQUE(OIB_studenta,  Sluzbeni_mail_studenta,
JMBAG)
);
...
CREATE TABLE Podaci_transfer(
    Podaci_transfer_ID INT IDENTITY(1,1),
    Naziv_tablice nvarchar(32) NOT NULL,
    Vrijednosti nvarchar(1000),
    CONSTRAINT Podaci_transfer_PK PRIMARY KEY (Podaci_transfer_ID)
);
...
```

Ispis 8: Dio skripte napravi_tablice.sql za različite baze

Za punjenje tablica podacima korištena je skripta *napuni_tablice.sql* koja se može vidjeti u ispisu 9.

```
INSERT INTO Student(Student_ID, Ime_studenta, Prezime_studenta, Datum_rodenja_studenta, Mjesto_rodenja_studenta, OIB_studenta, Spol_studenta, Broj_telefona_studenta, JMBAG, Sluzbeni_mail_studenta, Smjer, Status_studenta)

VALUES (1,'Ana', 'Anić', '2000-01-01', 'Split, Hrvatska', 123456789, 'Ž', '+385977787474', 0008521 47, 'aanic@unist.hr', 'RAČ', 'Redovni'),

(2, 'Ivo', 'Ivić', '1999-05-05', 'Osijek, Hrvatska', 987654321, 'M', '+385912356474', 400500123, 'iivic@unist.hr', 'RAČ', 'Izvanredni'),

(3, 'Roko', 'Grubić', '1996-07-21', 'Zadar, Hrvatska', 60224282570, 'M', '+385977824156', 0009075115, 'rg47947@unist.hr', 'RAČ', 'Redovni');

...
```

Ispis 9: Dio skripte napuni_tablice.sql za različite baze

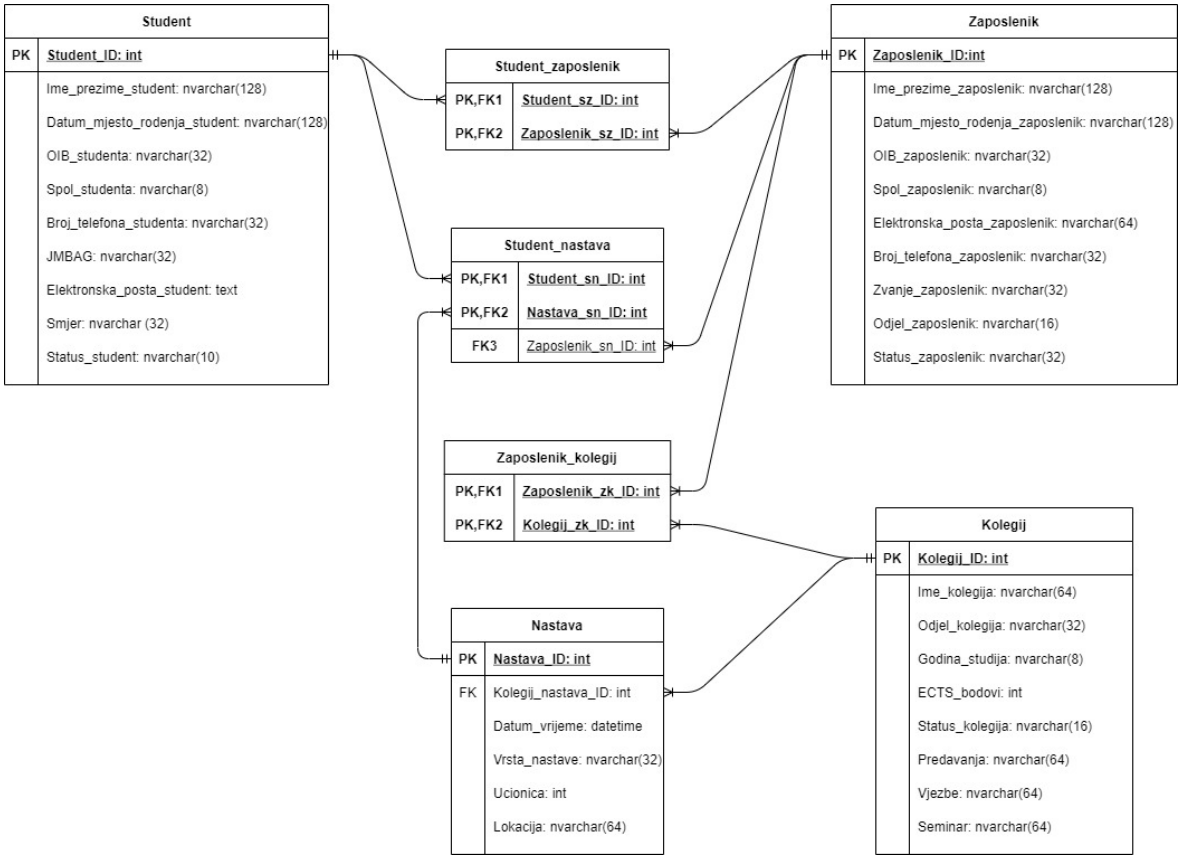
Kako bi se dvije prethodne skripte enkapsulirale unutar PS-a, kreirana je skripta *inicijaliziraj_bazu.ps1* koja se može vidjeti u ispisu 10.

```
$server = "DESKTOP-17PP6LU\SQLEXPRESS"

$db = "nastava_db"

Invoke-Sqlcmd -ServerInstance $server -Query "CREATE DATABASE $db ON (NAME = nastava_db_data,

FILENAME = 'E:\SQL Express\MSSQL15.SQLEXPRESS\MSSQL\DATA\nastava_db_data.mdf', SIZE = 8MB, MAXSIZE = 6GB, FILEGROWTH = 5MB)

LOG ON (NAME = nastava_db_log,

FILENAME = 'E:\SQL Express\MSSQL15.SQLEXPRESS\MSSQL\DATA\nastava_db_log.ldf', SIZE = 4MB, MAXSIZE = 3GB, FILEGROWTH = 5MB);"

Invoke-Sqlcmd -ServerInstance $server -Database $db -Query "USE $db; ALTER DATABASE $db SET RECOVERY FULL;"

Invoke-Sqlcmd -ServerInstance $server -Database $db -InputFile "E:\Zavrsni_ispit\Skripte_2\SQL\napravi_tablice.sql"

Invoke-Sqlcmd -ServerInstance $server -Database $db -InputFile "E:\Zavrsni_ispit\Skripte_2\SQL\napuni_tablice.sql"
```

Ispis 10: Skripta inicijaliziraj_bazu.ps1 za različite baze

Nakon izvršenja skripte u PS-u napravljena je baza podataka, njene tablice i umetnuti su podaci unutar tablica. Objašnjenje kôda je isto kao i kod identičnih bâza budući da se naredbe ne mjenjaju, nego samo broj i tip podataka te putanje do skripti.

Baza podataka *kolegiji_oss* modelirana je drukčije od baze *nastava_db*, ali također sadrži podatke o praćenju nastave na fakultetu. Može se zamisliti primjer gdje aplikacija referade koristi jednu, a studentska aplikacija za praćenje nastave drugu bazu podataka. Isto tako, moguće je da te baze nemaju identičnu strukturu i nazivlje, ali su isti podaci potrebni i u jednoj i u drugoj bazi te ih treba međusobno sinkronizirati. Dijagram entiteti-veze prikazan je na slici 7.



Slika 7: Model baze podataka kolegiji_oss

U usporedbi sa bazom podataka *nastava_db* vidi se razlika u tipovima podataka kao i spajanje nekih atributa (ime i prezime, datum i mjesto rođenja i sl.), a tablica Ucionica je „ubačena“ unutar tablice Nastava. Relacije između tablica su tipa jedan-na-više, a model podataka je prikazan u nastavku.

Entitet Student prikazan je u tablici 17.

Tablica 17: Tablica Student

Student			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Student_ID	int	/	(1,1)
Ime_prezime_studenta	nvarchar	128	(1,1)
Datum_mjesto_rodenja_studenta	nvarchar	128	(1,1)
OIB_studenta	nvarchar	32	(1,1)
Spol_studenta	nvarchar	8	(1,1)
Broj_telefona_studenta	nvarchar	32	(0,1)
JMBAG	nvarchar	32	(1,1)
Elektronska_posta_student	nvarchar	64	(0,1)
Smjer	nvarchar	32	(1,1)
Status_studenta	nvarchar	10	(1,1)

Entitet Zaposlenik prikazan je u tablici 18.

Tablica 18: Tablica Zaposlenik

Zaposlenik			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Zaposlenik_ID	int	/	(1,1)
Ime_prezime_zaposlenik	nvarchar	128	(1,1)
Datum_mjesto_rodenja_zaposlenik	nvarchar	128	(1,1)
OIB_zaposlenik	nvarchar	32	(1,1)
Spol_zaposlenik	nvarchar	8	(1,1)
Elektronska_posta_zaposlenik	nvarchar	64	(0,1)
Broj_telefona_zaposlenik	nvarchar	32	(0,1)
Zvanje_zaposlenik	nvarchar	32	(1,1)
Odjel_zaposlenik	nvarchar	16	(1,1)
Status_zaposlenik	nvarchar	32	(1,1)

Entitet Kolegij je prikazan u tablici 19.

Tablica 19: Tablica Kolegij

Kolegij			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Kolegij_ID	int	/	(1,1)
Ime_kolegija	nvarchar	64	(1,1)
Odjel_kolegija	nvarchar	32	(1,1)
Godina_studija	nvarchar	8	(1,1)
ECTS_bodovi	int	/	(1,1)
Status_kolegija	nvarchar	16	(1,1)
Predavanja	nvarchar	64	(0,1)
Vjezbe	nvarchar	64	(0,1)
Seminar	nvarchar	64	(0,1)

Entitet Nastava prikazan je u tablici 20.

Tablica 20: Tablica Nastava

Nastava			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Termin_ID	int	/	(1,1)
(FK) Kolegij_nastava_ID	int	/	(1,1)
Datum_vrijeme	datetime	/	(0,1)
Vrsta_nastave	nvarchar	32	(0,1)
Ucionica	int	/	(0,1)
Lokacija	nvarchar	64	(0,1)

Entitet Student_zaposlenik prikazan je u tablici 21.

Tablica 21: Student_zaposlenik

Student_zaposlenik			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) (FK) Student_sz_ID	int	/	(1,1)
(PK) (FK) Zaposlenik_sz_ID	int	/	(1,1)

Entitet Zaposlenik_kolegij prikazan je u tablici 22.

Tablica 22: Tablica Zaposlenik_kolegij

Zaposlenik_kolegij			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) (FK) Zaposlenik_zk_ID	int	/	(1,1)
(PK) (FK) Kolegij_zk_ID	int	/	(1,1)

Entitet Student_nastava prikazan je u tablici 23.

Tablica 23: Tablica Student_nastava

Student_nastava			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) (FK) Student_sn_ID	int	/	(1,1)
(PK) (FK) Nastava_sn_ID	int	/	(1,1)
(FK) Zaposlenik_sn_ID	int	/	(1,1)

SQL skripta *napravi_tablice.sql* vidljiva je u ispisu 11.

```
CREATE TABLE Student(
    Student_ID INT NOT NULL,
    Ime_prezime_studenta NVARCHAR(128) NOT NULL,
    Datum_mjesto_rodenja_studenta NVARCHAR(128) NOT NULL,
    OIB_studenta NVARCHAR(32) NOT NULL,
    Spol_studenta NVARCHAR(8) NOT NULL,
    Broj_telefona_studenta NVARCHAR(32),
    JMBAG NVARCHAR(32) NOT NULL,
    Elektronska_posta_student NVARCHAR(64),
    Smjer NVARCHAR(32) NOT NULL,
    Status_student NVARCHAR(10) NOT NULL,
    CONSTRAINT Student_PK PRIMARY KEY(Student_ID),
    CONSTRAINT Student_uq UNIQUE(OIB_studenta, Elektronska_posta_student,
JMBAG)
);
...
CREATE TABLE Podaci_transfer(
    Podaci_transfer_ID INT IDENTITY(1,1),
    Naziv_tablice nvarchar(32) NOT NULL,
    Vrijednosti nvarchar(1000),
    CONSTRAINT Podaci_transfer_PK PRIMARY KEY (Podaci_transfer_ID)
);
...
```

Ispis 11: Dio skripte napravi_tablice.sql za različite baze

Za punjenje tablica podacima korištena je skripta *napuni_tablice.sql* koja se može vidjeti u ispisu 12.

```
INSERT INTO Student(Student_ID, Ime_prezime_studenta,
Datum_mjesto_rodenja_studenta,
OIB_studenta, Spol_studenta, Broj_telefona_studenta, JMBAG,
Elektronska_posta_student, Smjer, Status_student)
VALUES (1,'Ana Anić', '2000-01-01, Split, Hrvatska', 123456789, 'Ž',
'+385977787474', 000852147, 'aanic@unist.hr', 'RAČ', 'Redovni'),
(2, 'Ivo Ivić', '1999-05-05, Osijek, Hrvatska', 987654321, 'M',
'+385912356474', 400500123, 'iivic@unist.hr', 'RAČ', 'Izvanredni'),
(3, 'Roko Grubić', '1996-07-21, Zadar, Hrvatska', 60224282570, 'M',
'+385977824156', 0009075115, 'rg47947@unist.hr', 'RAČ', 'Redovni');
```

Ispis 12: Dio skripte napuni_tablice.sql za različite baze

Kako bi se prethodne dvije skripte enkapsulirale unutar PS-a, kreirana je skripta *inicijaliziraj_bazu.ps1* koja se može vidjeti u ispisu 13.

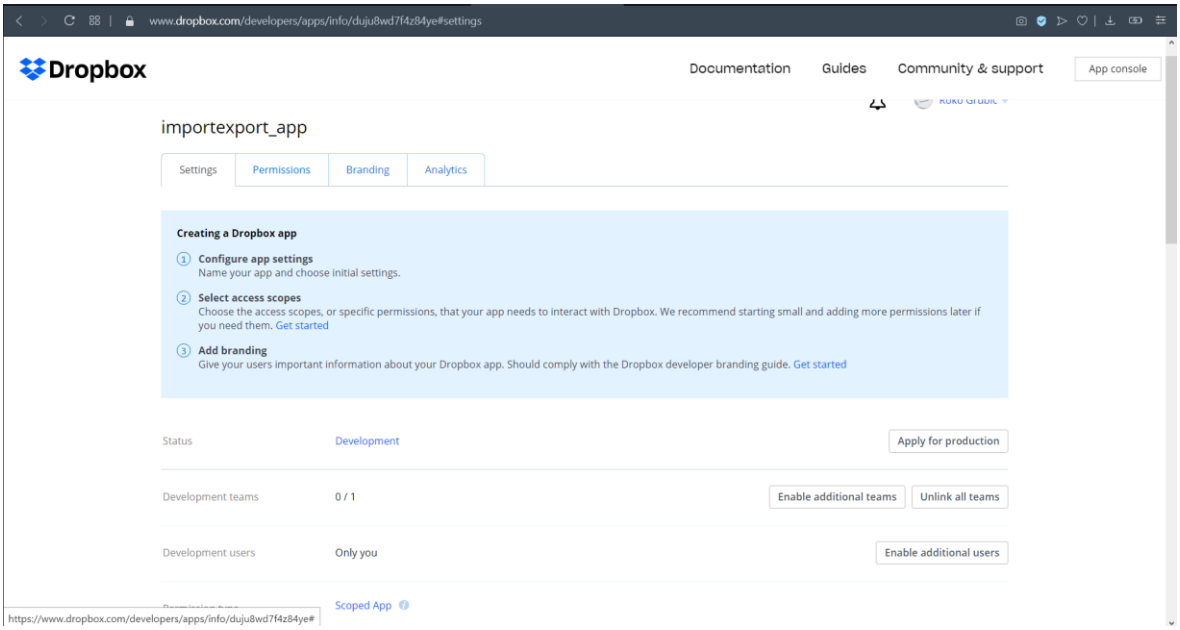
```
$server = "DESKTOP-D4UT8J3\SQLEXPRESS"

$db = "kolegiji_oss"

Invoke-Sqlcmd -ServerInstance $server -Query "CREATE DATABASE $db ON (NAME
= kolegiji_oss_data,

FILENAME         =         'C:\Program         Files\Microsoft         SQL
Server\MSSQL15.SQLEXPRESS\MSSQL\DATA\kolegiji_oss_data.mdf', SIZE = 8MB,
MAXSIZE = 6GB, FILEGROWTH = 5MB)

LOG ON (NAME = kolegiji_oss_log,

FILENAME         =         'C:\Program         Files\Microsoft         SQL
Server\MSSQL15.SQLEXPRESS\MSSQL\DATA\kolegiji_oss_log.ldf',  SIZE  =  4MB,
MAXSIZE = 3GB, FILEGROWTH = 5MB);"

Invoke-Sqlcmd -ServerInstance $server -Database $db -Query "USE $db; ALTER
DATABASE $db SET RECOVERY FULL;"

Invoke-Sqlcmd   -ServerInstance   $server   -Database   $db   -InputFile
"C:\Zavrsni_ispit\Skripte_2\SQL\napravi_tablice.sql"

Invoke-Sqlcmd   -ServerInstance   $server   -Database   $db   -InputFile
"C:\Zavrsni_ispit\Skripte_2\SQL\napuni_tablice.sql"
```

Ispis 13: Skripta inicijaliziraj_bazu.ps1 za različite baze

Nakon izvršenja skripte u PS-u napravljena je baza podataka, njene tablice i umetnuti su podaci unutar tablica. Objašnjenje kôda je isto kao i kod identičnih baza budući da se naredbe ne mjenjaju, nego samo broj i tip podataka te putanje do skripti.

3.3. Izrada Dropbox aplikacije za prijenos datoteka

Kako bi se datoteke prenosile bez „fizičkih“ medija (USB, tvrdi diskovi i slično) i kako bi se sinkronizacija mogla automatizirati odabran je Dropbox. Prijenos podataka je moguć uz račun na Dropboxu i kreiranje vlastite aplikacije kojoj se podese dopuštenja za postavljanje i preuzimanje datoteka kojima se upravlja. Aplikacija se izradi u nekoliko jednostavnih koraka kod kojih se odabere API (*Scoped access*), tip pristupa (*Full Dropbox*) i naziv aplikacije [12]. Nakon kreiranja aplikacije, unutar dopuštenja su odabrane opcije *files.content.write* i *files.content.read* koje dopuštaju skriptama postavljanje i preuzimanje podataka. *Access Token* je niz znakova i slova koji služi za autentifikaciju kod pristupa aplikaciji i može se generirati svaka 4 sata ili postaviti da neka od generiranih sekvenci traje zauvijek. Napravljena aplikacija se naziva *importexport_app*, a sučelje aplikacije je prikazano na slici 8.



Slika 8: importexport_app

3.4. Rješenje zadatka za identične baze

3.4.1. Export tablice i okidači

Jedno od rješenja zadatka sastoji se od dodavanja *export* tablica koje su identične tablicama u koje se spremaju podaci, a model podataka jedne od njih (ostale su izrađene po istom principu) može se vidjeti u tablici 24.

Tablica 24: Tablica Predmet_export

Predmet_export			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Promjena_predmet_ID	int	/	(1,1)
Tip_promjene_pre	varchar	16	(1,1)
Predmet_export_ID	int	/	(1,1)
Naziv_predmeta_export	varchar	32	(1,1)
Odjel_predmeta_export	varchar	32	(1,1)
Godina_studija_export	varchar	8	(1,1)
Broj_ECTSa_export	int	/	(1,1)
Status_predmeta_export	varchar	16	(1,1)
Predavanja_export	varchar	16	(0,1)
Vjezbe_export	varchar	32	(0,1)
Seminar_export	varchar	16	(0,1)

Tablice imaju svoj primarni ključ i polje za tip operacije nad redcima u bazi (insert, update ili delete), a napravljene su unutar *napravi_tablice.sql* skripte.

Okidač je spremljena procedura (engl. *stored procedure*) u SQL bazi podataka koja se automatski aktivira nakon određenog događaja i ima unaprijed određenu sintaksu koju je potrebno slijediti [13]. U završnom radu okidači su korišteni nakon svakog umetanja, ažuriranja ili brisanja redaka u bazi, a kôd skripte pod nazivom *svi_okidaci.sql* prikazan je u ispisu 14.

```
...
CREATE TRIGGER Ucionica_export_trigger
ON Ucionica
AFTER INSERT, DELETE, UPDATE
AS
IF EXISTS(SELECT 1 from inserted) AND EXISTS(SELECT 1 from deleted)
BEGIN
    INSERT   INTO   Ucionica_export(Tip_promjene_u,   Ucionica_export_ID,
Broj_ucionice_export,
    Lokacija_export, Namjena_export)
    SELECT 'UPDATE', Ucionica_ID, Broj_ucionice, Lokacija, Namjena from
inserted ORDER BY Ucionica_ID ASC;
END
IF EXISTS(SELECT 1 from inserted) AND NOT EXISTS(SELECT 1 from deleted)
BEGIN
    INSERT   INTO   Ucionica_export(Tip_promjene_u,   Ucionica_export_ID,
Broj_ucionice_export,
    Lokacija_export, Namjena_export)
    SELECT 'INSERT', Ucionica_ID, Broj_ucionice, Lokacija, Namjena from
inserted ORDER BY Ucionica_ID ASC;
END
IF NOT EXISTS(SELECT 1 from inserted) AND EXISTS(SELECT 1 from deleted)
BEGIN
    INSERT   INTO   Ucionica_export(Tip_promjene_u,   Ucionica_export_ID,
Broj_ucionice_export,
    Lokacija_export, Namjena_export)
    SELECT 'DELETE', Ucionica_ID, Broj_ucionice, Lokacija, Namjena from
deleted ORDER BY Ucionica_ID ASC;
END
GO
...
```

Ispis 14: Dio skripte svi_okidaci.sql

`CREATE TRIGGER ime_okidača ON ime_tablice AFTER INSERT, DELETE, UPDATE AS` je sintaksa za kreiranje okidača koji se aktivira nakon umetanja, ažuriranja ili brisanja redaka iz tablica. Bitno je spomenuti `inserted` i `deleted` tablice koje dolaze u „sklopu“ okidača i u njima se nalaze umetnuti i izbrisani redci. Logikom `IF` uvjeta *export* tablice pune se podacima koji su izbrisani ili dodani uz tip operacije koja je izvršena. Kako u pozadini ažuriranja tablice stoji da se promjenjeni redak obriše pa naknadno doda na isto mjesto u bazi, rješen je problem ažuriranih redaka. Ako se redak nalazi samo u `inserted`

tablici tada je tip operacije `INSERT`, ako se nalazi samo u `deleted` tablici tip operacije je `DELETE`, a ako se nalazi u obje tablice tada se radi `UPDATE`.

Srž ovog završnog rada su transfer datoteke tipa CSV, XML i JSON pa su tako napravljene PS skripte za izvoz podataka u sva tri tipa datoteka. *Export_to_csv.ps1* služi za izvoz podataka u CSV datoteku i može se vidjeti u ispisu 15.

```
$server = "DESKTOP-17PP6LU\SQLEXPRESS"
$db = "pracenje_nastave_db"


$csv_putanja = "E:\Zavrsni_ispit\Datoteke_triggers\Csv_datoteke\"
if (!(Test-Path $csv_putanja)) {
    mkdir $csv_putanja
}
else {
    foreach ($file in Get-ChildItem $csv_putanja) {
        Remove-Item
"E:\Zavrsni_ispit\Datoteke_triggers\Csv_datoteke\$file"
    }
}


$promjene_student = Invoke-Sqlcmd -ServerInstance $server -Database $db -
Query "SELECT * FROM Student_export;"
if ($promjene_student) {
    $promjene_student    |    Export-Csv    -NoTypeInformation    -Path
"E:\Zavrsni_ispit\Datoteke_triggers\Csv_datoteke\Promjene_student.csv"  -
Encoding UTF8
    Write-Host -ForegroundColor Green "Promjene u tablici Student su
zapisane u csv datoteku! `n"
}
else {
    Write-Host -ForegroundColor Red "Nema promjena u tablici Student, csv
datoteka nije napravljena! `n"
}
...
```

Ispis 15: Dio skripte export_to_csv.ps1

Prvo se provjerava postoji li putanja do mape na računalu u koju se spremaju datoteke i ako mapa ne postoji izrađuje se naredbom `mkdir $putanja`. Ukoliko mapa ipak postoji sa `foreach` petljom se prolazi kroz mapu i brišu se stare datoteke kako ne bi došlo do sukoba

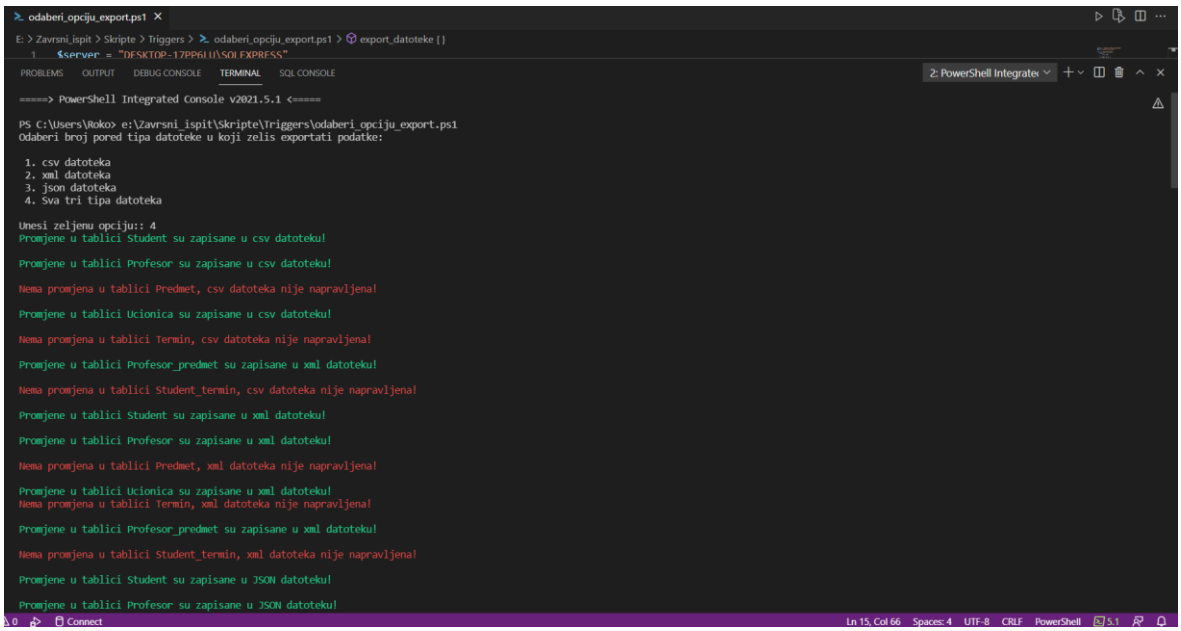
podataka. Nakon toga se preko `Invoke-Sqlcmd` naredbe pristupa bazi i biraju se svi redci iz *export* tablica preko `SELECT * FROM ime_tablice` te se rezultat sprema u varijablu. Ukoliko varijabla nije prazna (postoje redci u tablicama) preko cjevovoda (oznaka „|“) kreće se u izvoz podataka preko predinstalirane `Export-Csv` naredbe. Naredba stvara CSV datoteku i za parametre prima putanju do datoteke (`-Path`) uz napomenu da unutar datoteke ne stavlja informaciju o tipu podataka (`-NoTypeInformation`). Datoteka se kodira UTF-8 standardom parametrom `-Encoding` i ispisuje se poruka o uspješnom kreiranju datoteke. Ukoliko redci ne postoje ispisuje se upozorenje i kreće se na slijedeću tablicu. Skripta *export_to_xml.ps1* služi za izvoz podataka u XML datoteku. Naredba za izvoz je u ovom slučaju `Export-Clixml` koja stvara XML datoteku i prima parametre za putanju (`-Path`) i kodiranje (`-Encoding`). *Export_to_json.ps1* koristi naredbu `ConvertTo-Json` koja stvara JSON datoteku i sastoji se od više svojstava nego što je potrebno za transfer podataka. Parametrom `-ExcludeProperty` isključena su „nepotrebna“ svojstva, a sa `Out-File` parametrom definirana je putanja do mape u koju se spremaju podaci [14]. Logika je ista u sva tri slučaja i objašnjena je na primjeru *export_to_csv.ps1* skripte.

Kako bi korisnik imao interakciju s događajima kod izvoza, napravljena je skripta *odaberi_opciju_export.ps1* koja se sastoji od više funkcija, a kôd je prikazan u ispisu 16.

```
$server = "DESKTOP-17PP6LU\SQLEXPRESS"
$db = "pracenje_nastave_db"
function export_datoteke {
    param ([int] $opt)
    if ($opt -eq 1) {
        .("E:\Zavrsni_ispit\Skripte\Triggers\export_to_csv.ps1") }
...
function upload_dropbox {
    Param([string]$SourceFilePath,
          [string]$TargetFilePath,
          [string]$AccessToken)
...
Invoke-RestMethod  -Uri  https://content.dropboxapi.com/2/files/upload  -
Method Post -InFile $SourceFilePath -Headers $headers }
...
function prodi_direktorij {
    param ([int]$opt)
    $access_token                                                    =
"bve5clQAe4sAAAAAAAAAAZjOMSuAugUUmCdvNILy9WxjYM8JSlOBSo_SOjRkNbAJ"
    if ($opt -eq 1) {
        $directory = "E:\Zavrsni_ispit\Datoteke_triggers\Csv_datoteke\"
        foreach ($file in Get-ChildItem $directory) {
            $izvorisni_csv                                           =
"E:\Zavrsni_ispit\Datoteke_triggers\Csv_datoteke\$file"
            $odredisni_csv = "/Datoteke_triggers/Csv_datoteke/$file"
            upload_dropbox -SourceFilePath $izvorisni_csv -TargetFilePath
$odredisni_csv -AccessToken $access_token
        } }
...
function main {
    Write-Host "Odaberi broj pored tipa datoteke u koji zelis exportati
podatke:"
    Write-Host " `n 1. csv datoteka `n 2. xml datoteka `n 3. json datoteka
`n 4. Sva tri tipa datoteka `n"
    $opcija = Read-Host -Prompt "Unesi zeljenu opciju:"
    export_datoteke -opt $opcija
    prodi_direktorij -opt $opcija }
...
```

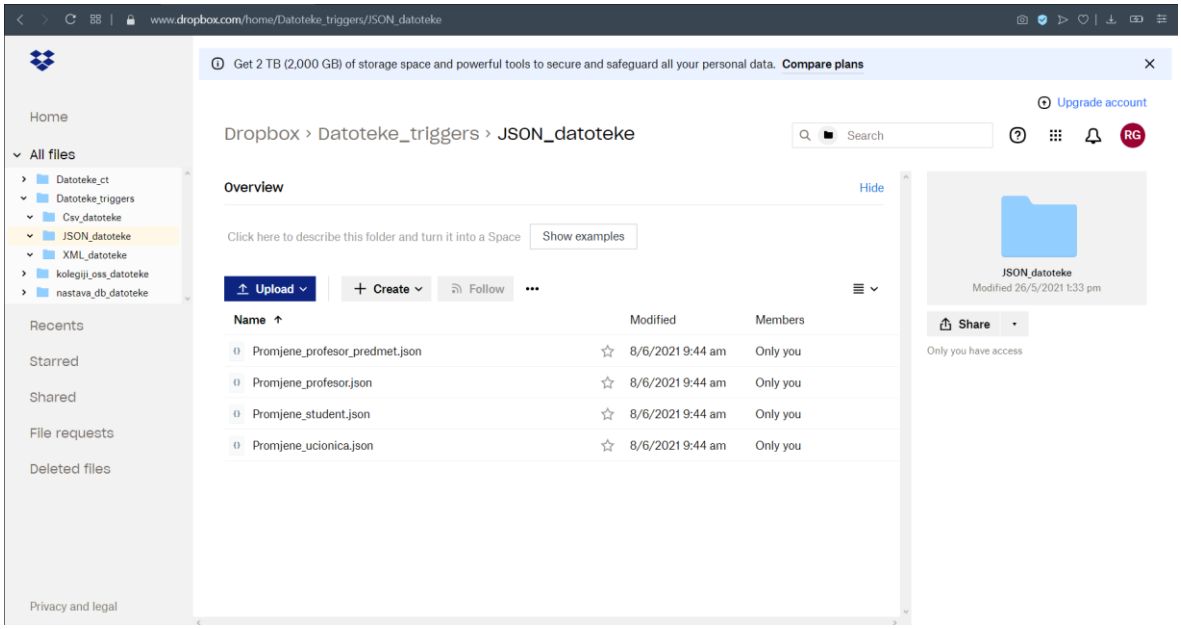
Ispis 16: Dio skripte odaberi_opciju_export.ps1

Funkcija `main` služi za ispis opcija izvoza i korisnički unos te pozivanje ostalih funkcija. `Export_datoteke` na temelju unosa korisnika (brojevi od 1 do 4; želi li korisnik izvoz u neku specifičnu ili sve vrste datoteka) poziva skripte za izvoz. Na kraju se podaci brišu iz *export* tablica kako ne bi zauzimale prostor budući da će podaci biti spremljeni u datoteke. `Prodi_direktorij` `forach` petljom prolazi kroz svaki od direktorija i poziva `upload_dropbox` funkciju koja za argumente prima izvorišnu putanju na računalu, odredišnu putanju na Dropbox računu i već spomenuti `Access Token`. Unutar funkcije se stvara zaglavlje u koje se dodaju parametri potrebni za autorizaciju i komunikaciju sa Dropbox API-em. Naredbom `Invoke-RestMethod` uz parametre `-Uri` (definira internetski resurs kojem funkcija pristupa), `-Method` (HTTP metoda za rad sa podacima na internetu), `-Infile` (određuje datoteku koju prenosimo) i `-Headers` (za zaglavlje) datoteka se postavlja na Dropbox [15]. Zaslon koji se pojavljuje pri izvršavanju skripte prikazan je na slici 9.



Slika 9: Izvršavanje skripte odaberi_opciju_export.ps1

Postavljene datoteke nalaze se na Dropboxu, a mogu se vidjeti na slici 10.



Slika 10: Postavljene datoteke na Dropboxu

Nakon što su datoteke postavljene, na drugom se računalu može pokrenuti uvoz podataka. Kako se i uvoz može napraviti iz CSV, XML i JSON datoteka, za tu svrhu napravljene su PS skripte. Prva skripta, naziva *import_from_csv.ps1*, prikazana je u ispisu 17.

```
... try {

    $csv_predmet              =              Import-Csv              -Path
"E:\Zavrsni_ispit\Datoteke_triggers\Csv_datoteke\Promjene_predmet.csv"

    Invoke-Sqlcmd -ServerInstance $server -Database $db -Query "DISABLE
TRIGGER Predmet_export_trigger ON Predmet;"

    $csv_predmet | ForEach-Object {

        if (_.Tip_promjene_pra -eq 'INSERT') {

Invoke-Sqlcmd -ServerInstance $server -Database $db -Query "INSERT INTO
Predmet values(CAST('$($_.Predmet_export_ID)' AS INT),

CAST('$($_.Naziv_predmeta_export)             AS             VARCHAR(32)),
CAST('$($_.Odjel_predmeta_export)' AS VARCHAR(32)),

CAST('$($_.Godina_studija_export)'             AS             VARCHAR(8)),
CAST('$($_.Broj_ECTSa_export)' AS INT),

CAST('$($_.Status_predmeta_export)'            AS            VARCHAR(16)),
CAST('$($_.Predavanja_export)' AS VARCHAR(16)),

CAST('$($_.Vjezbe_export)'  AS  VARCHAR(32)),  CAST('$($_.Seminar_export)'
AS VARCHAR(16)));" }

        elseif ($_.Tip_promjene_pra -eq 'UPDATE') {

            Invoke-Sqlcmd  -ServerInstance  $server  -Database  $db  -Query
"UPDATE Predmet SET

            Predmet_ID   =   CAST('$($_.Predmet_export_ID)'   AS   INT),
Naziv_predmeta = CAST('$($_.Naziv_predmeta_export)' AS VARCHAR(32)),

            Odjel_predmeta   =   CAST('$($_.Odjel_predmeta_export)'   AS
VARCHAR(32)),  Godina_studija  =  CAST('$($_.Godina_studija_export)'  AS
VARCHAR(8)),

            Broj_ECTSa   =   CAST('$($_.Broj_ECTSa_export)'   AS   INT),
Status_predmeta = CAST('$($_.Status_predmeta_export)' AS VARCHAR(16)),

            Predavanja = CAST('$($_.Predavanja_export)' AS VARCHAR(16)),
Vjezbe = CAST('$($_.Vjezbe_export)' AS VARCHAR(32)),

            Seminar = CAST('$($_.Seminar_export)' AS VARCHAR(16)) WHERE
Predmet_ID = CAST('$($_.Predmet_export_ID)' AS INT);" }

        else {

            Invoke-Sqlcmd    -ServerInstance    $server_name    -Database
$database -Query `

"DELETE FROM Predmet WHERE Predmet_ID = CAST('$($_.Predmet_export_ID)' AS
INT)" } }

    Invoke-Sqlcmd -ServerInstance $server -Database $db -Query "ENABLE
TRIGGER Predmet_export_trigger ON Predmet;"

    Write-Host  -ForegroundColor  Green  "Tablica  Predmet  je  uspjesno
azurirana! `n" }

catch {

Write-Warning $Error[0] } ...
```

Ispis 17: Dio skripte import_from_csv.ps1

Postavljanjem `$ErrorActionPreference = „Stop“` spriječava se prerani prekid izvođenja. Unutar `try` i `catch` iznimke („pokušaj napraviti ono što piše u `try`, ako ne uspije uhvati pogrešku u `catch`“) odvija se sinkronizacija podataka na udaljenoj bazi. Funkcijom `Import-Csv` uz parametar `-Path` u varijablu se sprema CSV datoteka napravljena u izvozu i kao takva ona predstavlja skup objekata (svojstvo PS-a gdje je sve objekt). Naravno, ukoliko se tablica nije mjenjala na prvoj bazi podataka, datoteka neće biti kreirana i unutar `catch` dijela biti će ispisano upozorenje. Unutar *export* tablica dodano je polje `Tip_promjene` gdje se preko okidača zapisuje operacija. Kada je datoteka učitana u varijablu isključuje se okidač na bazi kako ne bi pokrenuo unos podataka za ažuriranje u *export* tablice naredbom `DISABLE TRIGGER ime_okidača ON ime_tablice`. Zatim se korištenjem cjevovoda i `ForEach-Object` petlje iterira preko objekata.

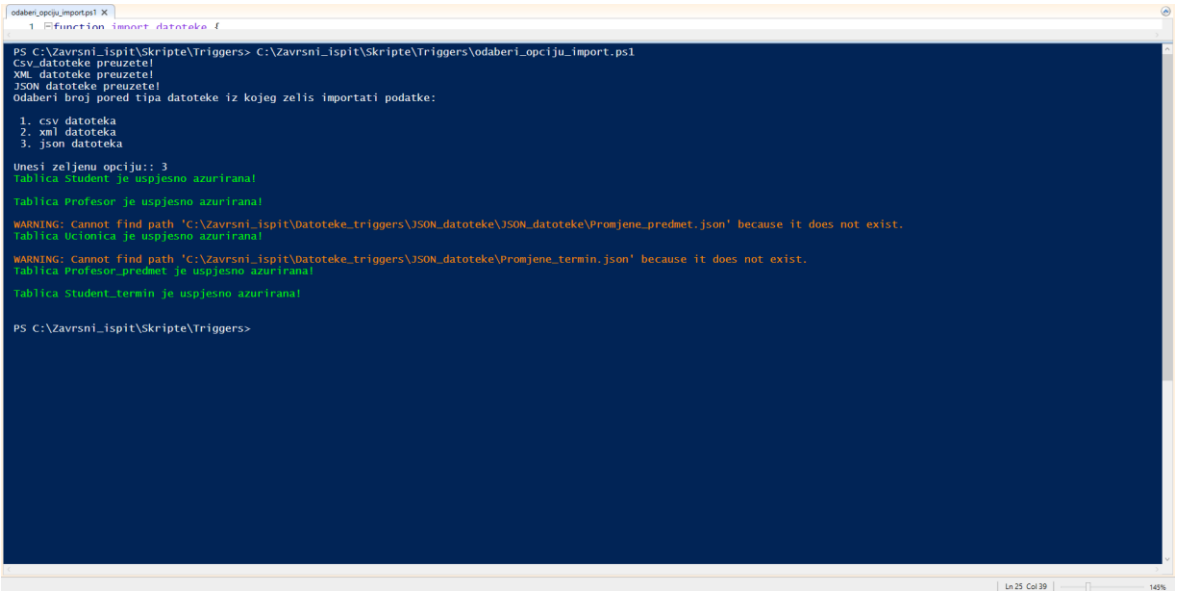
Ukoliko je polje `Tip_promjene` jednako „`INSERT`“ tada se sa `Invoke-Sqlcmd` funkcijom obavlja umetanje podataka sa naredbom `INSERT INTO naziv_tablice`. Kod kreiranja CSV, XML i JSON datoteka može se dogoditi promjena tipa podataka (npr. svi podaci postanu stringovi). Stoga je potrebno napraviti pretvorbu tipova podataka kako bi odgovarali bazi. SQL Server 2019 ima u sebi ugrađenu funkciju da samostalno pretvara podatke kako bi odgovarali bazi, ali to dovodi do slabijih performansi i mogućih neželjenih grešaka. Pretvorba je napravljena tako da je svaki podatak stavljen unutar `CAST` SQL funkcije koja pretvara podatke u željeni tip. Nadalje, ukoliko je polje `Tip_promjene` jednako „`UPDATE`“ obavlja se ažuriranje preko naredbe `UPDATE ime_tablice SET ime_atributa = nova_vrijednost WHERE`. Tu je također upotrijebljena funkcija `CAST` za pretvorbu podataka u odgovarajući tip. Za kraj, ukoliko `Tip_promjene` nije „`INSERT`“ ni „`UPDATE`“ definitivno je „`DELETE`“ i brisanje redaka se obavlja naredbom `DELETE FROM ime_tablice WHERE primarni_kljuc = primarni_kljuc_export_tablice`. Ovakav postupak se izvodi za svaku od tablica u bazi i obavlja sinkronizaciju podataka. *Import_from_xml.ps1* je naziv skripte koja služi za uvoz podataka iz XML datoteke. Koristi se funkcija `Import-Clixml` uz navođenje putanje (`-Path`) koja sprema datoteku u varijablu. Kod skripte *import_from_json.ps1* koristi se funkcija `Get-Content` uz parametar `-Path` koja se preko cjevovoda i funkcije `ConvertFrom-Json` sprema u varijablu. Logika je ista za sva tri slučaja, a objašnjena je na primjeru skripte *import_from_csv.ps1*.

Korisnik može odabrati iz koje vrste datoteka želi sinkronizirati bazu pa je u tu svrhu napravljena skripta naziva odaberi *odaberi_opciju_import.ps1* čiji se kôd može vidjeti u ispisu 18.

```
function import_datoteke {
    param ([int] $opt)
    if ($opt -eq 1) {
        .("E:\Zavrsni_ispit\Skripte\Triggers\import_from_csv.ps1")
    } ...
function skini_datoteke {
    [CmdletBinding()]
    Param([string]$DropboxFolder, [string]$OutputFolder) ...
Invoke-RestMethod `
        -Method POST `
        -Uri "https://content.dropboxapi.com/2/files/download_zip" `
        -Headers $headers `
        -OutFile $Tempfolder -ContentType ""...
function ukloni_datoteke {...
    foreach ($i in $lista_csv) {
        try {
            Invoke-RestMethod `
                -Method POST `
                -Uri "https://api.dropboxapi.com/2/files/delete_v2" `
                -Headers $headers `
                -Body $i}
        catch {
            Write-Warning $Error[0] }
    }...
function main {
    try {
        $mapa_dropbox = "/Datoteke_triggers/Csv_datoteke"
        $mapa_pc = "E:\Zavrsni_ispit\Datoteke_triggers"
        skini_datoteke    -DropboxFolder    $mapa_dropbox    -OutputFolder
$mapa_pc
        Write-Host "Csv_datoteke preuzete!" }
    catch {
        Write-Warning $Error[0] }...
```

Ispis 18: Dio skripte odaberi_opciju_import.ps1

Prvo se unutar funkcije `main` preko `try` i `catch` iznimke datoteke preuzimaju uz pomoć funkcije `skini_datoteke`. `Skini_datoteke` pristupa Dropboxu preko API-a, a za argumente prima putanju do mape na računalu i putanju do mape na Dropboxu. Dodaju se varijable za autorizaciju i zaglavlje kao u funkciji za postavljanje datoteka. `$Tempfolder` varijabla služi za spremanje datoteka u privremenu mapu na lokalnom računalu, a preko `Invoke-RestMethod` funkcije se datoteke preuzimaju. Argumenti su `-Method` (u ovom slučaju metoda je *Post*), `-Uri` označava stranicu kojoj se pristupa, `-Headers` označava zaglavlje, a `-OutFile` predstavlja putanju do mape za preuzimanje. Kako su datoteke u *zip* obliku, preko `Expand-Archive` funkcije podaci se „otpakiraju“ na željeno mjesto i sa `Remove-Item` privremena mapa se briše kako ne bi zauzimala prostor [16]. Nakon toga se u funkciji `main` ispisuju opcije za import datoteke i poziva se funkcija `import_datoteke`. Kao argumente prima odabranu korisničku opciju i poziva skripte za izvoz. Za kraj, bitno je razmisliti i o mogućem sukobu podataka. Primjer toga je slučaj kada se u prvoj sinkronizaciji ažuriraju sve tablice i kreirane su sve datoteke. One se nalaze na Dropboxu i preuzimaju se sa drugog računala. Drugom sinkronizacijom ažurirane su dvije tablice i dvije datoteke „prepišu“ (engl. *overwrite*) prethodno kreirane međutim ostale datoteke ostaju na Dropboxu i ponovno se radi sinkronizacija sa istim podacima. Rješenje je predstavljeno u obliku funkcije `ukloni_datoteke`. Unutar funkcije se rade liste svih datoteka čiji su članovi putanje do datoteka na Dropboxu u JSON formatu. Uz standardnu autorizaciju i potrebne argumente za pristup opisane u funkciji `skini_datoteke` preko petlje `foreach` prolazi se kroz listu i datoteke se brišu sa računa te se uklanja mogućnost sinkronizacije baze sa starim podacima [17]. Izvršavanjem ove skripte, sinkronizacija podataka je obavljena, a zaslon koji se izvršava pri pokretanju skripte vidi se na slici 11.



Slika 11: Izvršavanje skripte odaberi_opciju_import.ps1

Po istom principu i sa istim skriptama sinkronizacija se obavlja i na drugom SQL poslužitelju uz izmjenu putanja do datoteka i promjene naziva samog poslužitelja.

3.4.2. SQL Server Change Tracking

SQL Server Change Tracking je ugrađeno rješenje za praćenje promjena na tablicama prvi put uvedeno u SQL Serveru 2008 i može biti postavljeno u svim SQL Server verzijama uključujući i Express verziju korištenu u završnom radu. Kada se omogući nad tablicama, Change Tracking stvara interne tablice naziva *Change_Tracking_<Object_ID>* i prati INSERT, UPDATE i DELETE operacije u bazi. Po definiciji, unutar tablica za praćenje Change Tracking sprema primarni ključ reda u tablici nad kojim je izvršena promjena uz tip operacije, ali bez vrijednosti koje su mijenjane. Stoga je bitno da svaka tablica ima definiran primarni ključ, a uz pomoć SQL-a moguće je u datoteke dodati vrijednosti koje su izmjenjene. Također, ovo rješenje ne sprema povijest promjena nego samo zadnje promjene pa je preko kôda potrebno omogućiti pravilnu sinkronizaciju podataka [18].

Omogućavanje rješenja napravljeno je preko SQL skripte naziva *enable_tracking.sql* čiji kôd se može vidjeti u ispisu 19.

```
USE master;
GO
ALTER DATABASE [pracenje_nastave_db]
SET CHANGE_TRACKING = ON
 (CHANGE_RETENTION = 2 DAYS, AUTO_CLEANUP = ON);

USE pracenje_nastave_db;
...
ALTER TABLE Predmet
ENABLE CHANGE_TRACKING
WITH (TRACK_COLUMNS_UPDATED = ON);
GO
...
```

Ispis 19: Dio skripte enable_tracking.sql

Naredbom `ALTER DATABASE ime_baze SET CHANGE_TRACKING = ON (CHANGE_RETENTION = 2 DAYS, AUTO_CLEANUP = ON);` Change Tracking omogućen je na razini cijele baze te se postavlja zadržavanje podataka 2 dana i automatsko čišćenje tablica. Nakon toga se naredbama `ALTER TABLE ime_tablice ENABLE CHANGE TRACKING WITH (TRACK_COLUMNS_UPDATE = ON);` postavlja nad svakom tablicom. Ova skripta je izvršena unutar skripte *inicijaliziraj_bazu.ps1* pri kreiranju baze podataka. Kako bi se

pristupilo nekoj od tablica koje su stvorene preko Change Trackinga potrebno je izvršiti naredbu `SELECT * FROM changetable (changes [ime_tablice], 0) as CT;`, a rezultati takvog upita nad tablicom Student vidljivi su u tablici 25.

Tablica 25: Izgled Change Tracking tablice

SYS_CHANGE_VERSION	SYS_CHANGE_CREATION_VERSION	SYS_CHANGE_OPERATION	SYS_CHANGE_COLUMNS	SYS_CHANGE_CONTEXT	STUDENT_ID
1	1	I	NULL	NULL	4

Na primjeru je vidljivo kako uz polja verzije izmjene redka (čija se vrijednost povećava kako se redak dalje modificira; npr. nakon unosa se ažurira ili obriše) postoje i polja vrste operacije (I – INSERT, U – UPDATE, D - DELETE) te primarni ključ reda koji je mijenjan. Također, nisu vidljiva polja koja su mijenjana, a u nastavku će biti objašnjeno kako su izvučena iz tablice i spremljena u datoteku.

Ct_export_to_json.ps1 služi za izvoz podataka u JSON datoteku i može se vidjeti u ispisu 20.

```
$server = "DESKTOP-17PP6LU\SQLEXPRESS"
$db = "pracenje_nastave_db"


$json_putanja = "E:\Zavrsni_ispit\Datoteke_ct\JSON_datoteke"
if (!(Test-Path $json_putanja)) {
    mkdir $json_putanja
}
else {
    foreach ($file in $json_putanja) {
        Remove-Item "E:\Zavrsni_ispit\Datoteke_ct\Csv_datoteke\$file"
    }
}
...
$promjene_profesor = Invoke-Sqlcmd -ServerInstance $server -Database $db
-Query "SELECT CT.SYS_CHANGE_OPERATION,
CT.Profesor_ID,             Ime_profesora,             Prezime_profesora,
Datum_rodenja_profesora, Mjesto_rodenja_profesora,
OIB_profesora,          Spol_profesora,          Sluzbeni_mail_profesora,
Broj_telefona_profesora, Zvanje, Maticno_uciliste,
Odjel_profesora,  Status_profesora,  Godina_zaposlenja  FROM  CHANGETABLE
(CHANGES [Profesor], 0) as CT LEFT JOIN [dbo].[Profesor] P
ON CT.Profesor_ID = P.Profesor_ID ORDER BY CT.Profesor_ID;"
if ($promjene_profesor) {
    $promjene_profesor  |  Select-Object  *  -ExcludeProperty  ItemArray,
Table, RowError, RowState, HasErrors |
    ConvertTo-Json                          |                        Out-File
"E:\Zavrsni_ispit\Datoteke_ct\JSON_datoteke\Promjene_profesor.json"    -
Encoding utf8
    Write-Host  -ForegroundColor  Green  "Promjene  u  tablici  Profesor  su
zapisane u JSON datoteku! `n"
}
else {
    Write-Host  -ForegroundColor  Red  "Nema  promjena  u  tablici  Profesor,
JSON datoteka nije napravljena! `n"
}
...
```

Ispis 20: Dio skripte ct_export_to_json.ps1

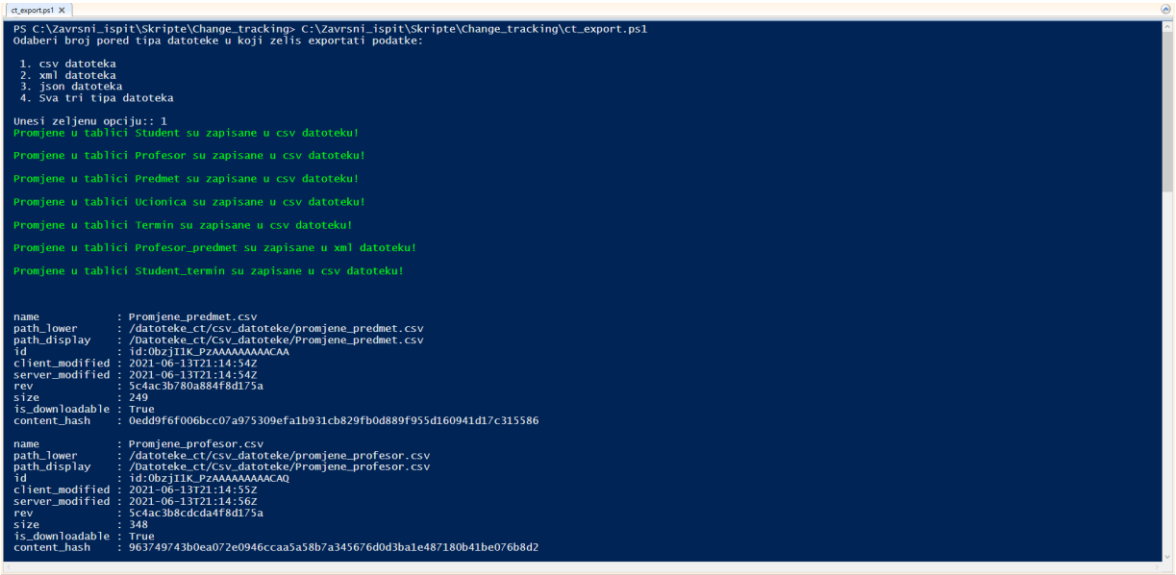
Prvo se provjerava postoji li putanja do mape na računalu u koju se spremaju datoteke i ako mapa ne postoji izrađuje se naredbom `mkdir $putanja`. Ukoliko mapa ipak postoji petljom `foreach` se prolazi kroz mapu i brišu se stare datoteke kako ne bi došlo do sukoba podataka. Nakon toga se preko naredbe `Invoke-Sqlcmd` pristupa bazi i biraju se redci vrste operacije i primarnog ključa redka upareni sa poljima iz originalnih tablica te se rezultat sprema u varijablu. Kako bi se dobili svi podaci potrebni za sinkronizaciju, pomoću SQL operatora `LEFT JOIN` spojene su Change Tracking tablice sa originalnim tablicama preko primarnog ključa. Ukoliko varijabla nije prazna (postoje redci u tablicama) preko cjevovoda (oznaka „|“) kreće se u izvoz podataka preko naredbe `ConvertTo-Json` koja stvara JSON datoteku i za parametre prima putanju do datoteke (`-Out-File`). Datoteka se kodira UTF-8 standardom preko parametra `-Encoding` i ispisuje se poruka o uspješnom kreiranju datoteke. Također, prije pretvorbe u JSON datoteku izostavljaju se „nepotrebna“ svojstva kako bi JSON format bio čist. Ukoliko redci ne postoje ispisuje se upozorenje i kreće se na slijedeću tablicu. Skripta *ct_export_to_xml.ps1* služi za izvoz podataka u XML datoteku. Naredba za izvoz je `Export-Clixml` koja stvara XML datoteku i prima parametre za putanju (`-Path`) i kodiranje (`-Encoding`). *Ct_export_to_csv.ps1* koristi naredbu `Export-Csv` koja stvara CSV datoteku. Parametrom `-Path` definirana je putanja do datoteke na računalu, a sa `-NoTypeInformation` u datoteku se ne sprema informacija o tipu podataka [14]. Logika je ista u sva tri slučaja i objašnjena je na primjeru skripte *ct_export_to_json.ps1*.

Da bi korisnik imao interakciju s događanjima kod izvoza napravljena je skripta *ct_export.ps1* koja se sastoji od više funkcija, a kôd je prikazan u ispisu 21.

```
function export_datoteke {
    param ([int] $opt)
    if ($opt -eq 1) {

.("E:\Zavrsni_ispit\Skripte\Change_tracking\ct_export_to_csv.ps1") }
...
function upload_dropbox {
    Param([string]$SourceFilePath,               [string]$TargetFilePath,
[string]$AccessToken )
...
Invoke-RestMethod  -Uri  https://content.dropboxapi.com/2/files/upload  -
Method Post -InFile $SourceFilePath -Headers $headers
...
function prodi_direktorij {
    param ([int]$opt)
    $access_token                                                          =
"bve5clQAe4sAAAAAAAAAAZjOMSuAugUUmCdvNILy9WxjYM8JSlOBSo_SOjRkNbAJ"
    if ($opt -eq 1) {
        $directory = "E:\Zavrsni_ispit\Datoteke_ct\Csv_datoteke\"
        foreach ($file in Get-ChildItem $directory) {
            $izvorisni_csv                                                 =
"E:\Zavrsni_ispit\Datoteke_ct\Csv_datoteke\$file"
            $odredisni_csv = "/Datoteke_ct/Csv_datoteke/$file"
            upload_dropbox -SourceFilePath $izvorisni_csv -TargetFilePath
$odredisni_csv -AccessToken $access_token }
}
...
function main {
    Write-Host "Odaberi broj pored tipa datoteke u koji zelis exportati
podatke:"
    Write-Host " `n 1. csv datoteka `n 2. xml datoteka `n 3. json datoteka
`n 4. Sva tri tipa datoteka `n"
    $opcija = Read-Host -Prompt "Unesi zeljenu opciju:"
    export_datoteke -opt $opcija
    prodi_direktorij -opt $opcija }
...
```

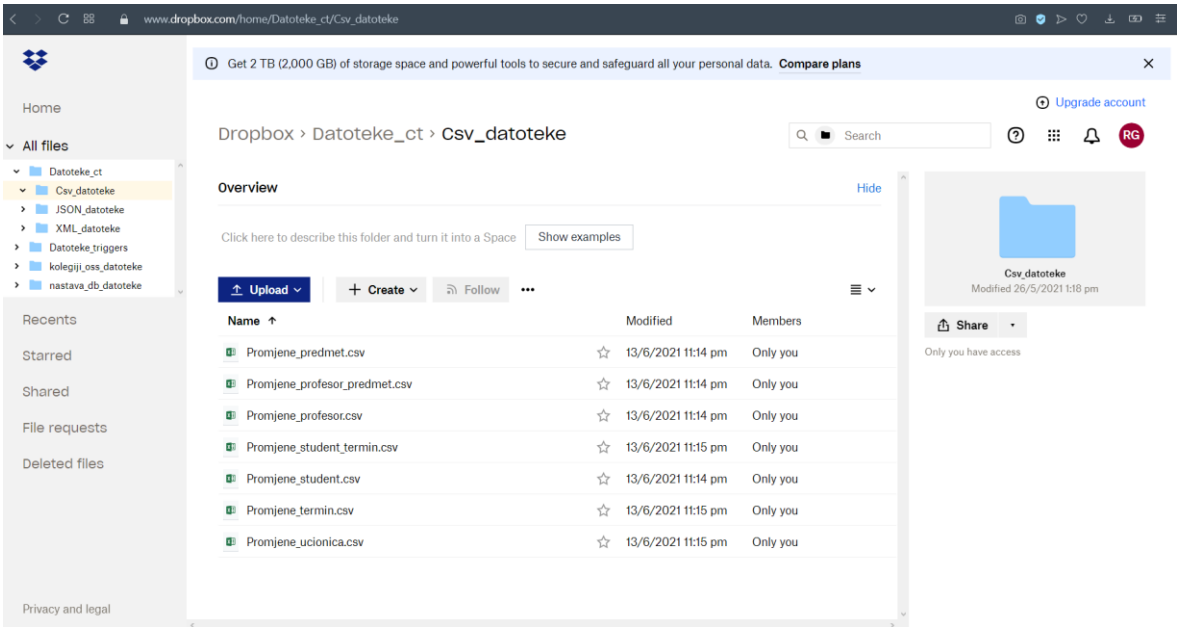
Ispis 21: Dio skripte ct_export.ps1

Funkcija `main` služi za ispis opcija izvoza i korisnički unos te pozivanje ostalih funkcija. `Export_datoteke` na temelju unosa (brojevi od 1 do 4; želi li korisnik izvoz u neku specifičnu ili sve vrste datoteka) poziva skripte za izvoz. Na kraju se podaci brišu iz *export* tablica kako ne bi zauzimale prostor budući da će podaci biti spremljeni u datoteke. `Prodi_direktorij` prolazi kroz svaki od direktorija i poziva funkciju `upload_dropbox` koja za argumente prima izvorišnu putanju na računalu, odredišnu putanju na Dropbox računu i već spomenuti `Access Token`. Unutar funkcije se stvara zaglavlje u koje se dodaju argumenti potrebni za autorizaciju i komunikaciju sa Dropbox API-em. Naredbom `Invoke-RestMethod` uz parametre `-Uri` (definira internetski resurs kojem funkcija pristupa), `-Method` (HTTP metoda za rad sa podacima na internetu), `-Infile` (određuje datoteku koju prenosimo) i `-Headers` (za zaglavlje) datoteka se postavlja na Dropbox [15]. Zaslon koji se pojavljuje pri izvršavanju skripte prikazan je na slici 12.



Slika 12: Izvršavanje skripte ct_export.ps1

Postavljene datoteke mogu se vidjeti na slici 13.



Slika 13: Datoteke na Dropbox računu

Nakon što su datoteke postavljene, na drugom računalu možemo pokrenuti uvoz podataka. Skripta naziva *ct_import_from_json.ps1* je prikazana u ispisu 22.

```
... try {

    $JSON_student_termin       =       Get-Content       -Raw       -Path
"E:\Zavrsni_ispit\Datoteke_ct\JSON_datoteke\Promjene_student_termin.json
" | ConvertFrom-Json

    $JSON_student_termin | ForEach-Object {

        $find_student_termin = Invoke-Sqlcmd -ServerInstance $server -
Database $db -Query `

 "Select Student_st_ID, Termin_st_ID from Student_termin WHERE

Student_st_ID    =    '$($_.Student_st_ID)'    AND    Termin_st_ID    =
'$($_.Termin_st_ID)'"

        if ($_.SYS_CHANGE_OPERATION -eq 'I') {

            if ($find_student_termin) {

                Invoke-Sqlcmd -ServerInstance $server -Database $db -Query

                    "DELETE  FROM  Student_termin  WHERE  Student_st_ID  =
CAST('$($_.Student_st_ID)'     AS     INT)     AND     Termin_st_ID     =
CAST('$($_.Termin_st_ID)' AS INT);"

                Invoke-Sqlcmd -ServerInstance $server -Database $db -Query
"INSERT INTO Student_termin

                values(CAST('$($_.Student_st_ID)'         AS         INT),
CAST('$($_.Termin_st_ID)' AS INT), CAST('$_.Profesor_st_ID' AS INT));" }

            else {

                Invoke-Sqlcmd -ServerInstance $server -Database $db -Query
"INSERT INTO Student_termin values (CAST('$($_.Student_st_ID)' AS INT),
CAST('$($_.Termin_st_ID)' AS INT), CAST('$_.Profesor_st_ID' AS INT));"} }

        elseif ($_.SYS_CHANGE_OPERATION -eq 'U') {

            if ($find_student_termin) {

                Invoke-Sqlcmd -ServerInstance $server -Database $db -Query
"UPDATE Student_termin SET Student_st_ID = CAST('$($_.Student_st_ID)' AS
INT), Termin_st_ID = CAST('$($_.Termin_st_ID)' AS INT) , Profesor_st_ID =
CAST('$($_.Profesor_st_ID)'     AS     INT)     WHERE     Student_st_ID     =
CAST('$($_.Student_st_ID)'     AS     INT)     AND     Termin_st_ID     =
CAST('$($_.Termin_st_ID)' AS INT);" }

            else {

                Invoke-Sqlcmd -ServerInstance $server -Database $db -Query
"INSERT INTO Student_termin values (CAST('$($_.Student_st_ID)' AS INT),
CAST('$($_.Termin_st_ID)' AS INT), CAST('$_.Profesor_st_ID' AS INT));"} }

        else {

            Invoke-Sqlcmd -ServerInstance $server -Database $db -Query `

                "DELETE   FROM   Student_termin   WHERE   Student_st_ID   =
CAST('$($_.Student_st_ID)'     AS     INT)     AND     Termin_st_ID     =
CAST('$($_.Termin_st_ID)' AS INT);" } }

    Write-Host -ForegroundColor Green "Tablica Student_termin je uspjesno
azurirana! `n" }

catch {

    Write-Warning $Error[0] } ...
```

Ispis 22: Dio skripte ct_import_from_json.ps1

Unutar `try` i `catch` iznimke odvija se sinkronizacija podataka na udaljenoj bazi. Naredbom `Get-Content` uz parametar `-Path` i funkcijom `ConvertFrom-Json` u varijablu se sprema JSON datoteka napravljena u izvozu datoteka i kao takva ona predstavlja skup objekata. Naravno, ukoliko se tablica nije mijenjala na prvoj bazi podataka, datoteka neće biti kreirana i unutar `catch` iznimke biti će ispisano upozorenje. Unutar Change Tracking tablica nalazi se polje `SYS_CHANGE_OPERATION` gdje je upisana vrsta operacije. Kada je datoteka učitana u varijablu preko cjevovoda i petlje `ForEach-Object` iterira se preko objekata.

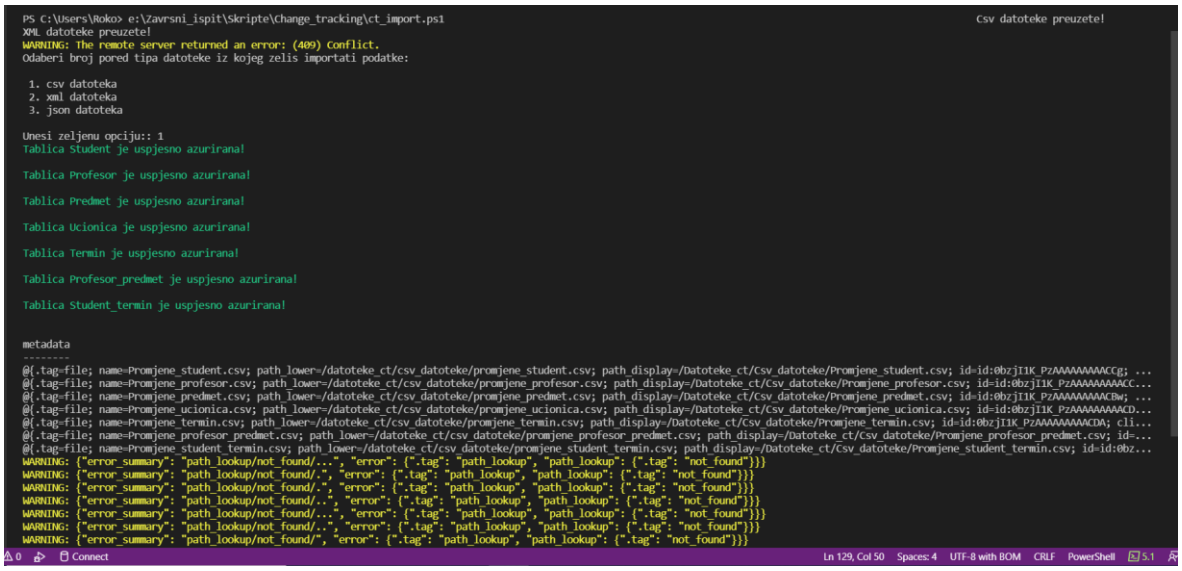
Ukoliko je polje `SYS_CHANGE_OPERATION` jednako „`I`“ tada se sa provjerava postoji li već redak sa istim primarnim ključem. Ukoliko postoji prvo se briše taj redak (zbog svojstva Change Trackinga da piše samo zadnje promjene nad redkom) pa se preko funkcije `Invoke-Sqlcmd` obavlja umetanje podataka sa naredbom `INSERT INTO naziv_tablice`. Kod kreiranja CSV, XML i JSON datoteka može se dogoditi promjena tipa podataka. Stoga je potrebno napraviti pretvorbu tipova podataka kako bi odgovarali bazi. Pretvorba je napravljena tako da je svaki podatak stavljen unutar `CAST` SQL funkcije koja pretvara podatke u željeni tip. Nadalje, ukoliko je polje `SYS_CHANGE_OPERATION` jednako „`U`“ ponovno se radi provjera postoji li redak. Ukoliko postoji, radi se ažuriranje preko naredbe `UPDATE ime_tablice SET ime_atributa = nova_vrijednost WHERE`. Tu je također upotrijebljena funkcija `CAST` za pretvorbu podataka u odgovarajući tip. U slučaju da redak ne postoji, obavlja se klasično umetanje redka. Za kraj, ukoliko Tip_promjene nije „`I`“ ni „`U`“ definitivno je „`D`“ i brisanje redaka se obavlja naredbom `DELETE FROM ime_tablice WHERE primarni_kljuc = primarni_kljuc_ct_tablice`. Ovakav postupak se izvodi za svaku od tablica u bazi i obavlja sinkronizaciju podataka. *Ct_import_from_xml.ps1* je naziv skripte koja služi za uvoz podataka iz XML datoteke. Koristi se funkcija `Import-Clixml` uz navođenje putanje (`-Path`) koja sprema datoteku u varijablu. Kod skripte *ct_import_from_csv.ps1* koristi se funkcija `Import-Csv` uz parametar `-Path` koja sprema podatke iz CSV datoteke u varijablu. Logika je ista za sva tri slučaja, a objašnjena je na primjeru skripte *import_from_json.ps1*.

Korisnik može odabrati iz koje vrste datoteka želi sinkronizirati bazu pa je u tu svrhu napravljena skripta naziva *ct_import.ps1* čiji se kôd može vidjeti u ispisu 23.

```
function import_datoteke {
    param (
        [int] $opt
    )
    if ($opt -eq 1) {

.("E:\Zavrsni_ispit\Skripte\Change_tracking\ct_import_from_csv.ps1")
    } ...
function skini_datoteke {
    [CmdletBinding()]
    Param( [string]$DropboxFolder, [string]$OutputFolder ) ...
    Invoke-RestMethod `
        -Method POST `
        -Uri "https://content.dropboxapi.com/2/files/download_zip" `
        -Headers $headers `
        -OutFile $Tempfolder -ContentType "" ...
function ukloni_datoteke { ...
foreach ($k in $lista_json) {
        try {
            Invoke-RestMethod `
                -Method POST `
                -Uri "https://api.dropboxapi.com/2/files/delete_v2" `
                -Headers $headers `
                -Body $k }
        catch {
            Write-Warning $Error[0] } } ...
function main { ...
try {
        $mapa_dropbox = "/Datoteke_ct/JSON_datoteke"
        $mapa_pc = "E:\Zavrsni_ispit\Datoteke_ct"
       skini_datoteke -DropboxFolder $mapa_dropbox -OutputFolder $mapa_pc
        Write-Host "JSON datoteke preuzete!"  }
    catch {
        Write-Warning $Error[0] }...
```

Ispis 23: Dio skripte ct_import.ps1

Prvo se unutar funkcije `main` preko `try` i `catch` iznimke datoteke preuzimaju uz pomoć funkcije `skini_datoteke`. `Skini_datoteke` pristupa Dropboxu preko API-a, a za argumente prima putanju do mape na računalu i putanju do mape na Dropboxu. Dodaju se varijable za autorizaciju i zaglavlje kao u funkciji za postavljanje datoteke. Varijabla `$Tempfolder` služi za spremanje datoteka u privremenu mapu na lokalnom računalu, a preko funkcije `Invoke-RestMethod` datoteke se preuzimaju. Argumenti su `-Method` (u ovom slučaju metoda je *Post*), `-Uri` označava stranicu kojoj se pristupa, `-Headers` označava zaglavlje, a `-OutFile` predstavlja putanju do mape za preuzimanje. Kako su datoteke u *zip* formatu, preko funkcije `Expand-Archive` podaci se „otpakiraju“ na željeno mjesto i sa `Remove-Item` privremena mapa se briše kako ne bi zauzimala prostor [16]. Nakon toga se u funkciji `main` ispisuju opcije za import datoteke i poziva se funkcija `import_datoteke`. Kao argumente prima odabranu korisničku opciju i poziva skripte za uvoz. Nakon uvoza podataka potrebno je ukloniti datoteke sa Dropboxa kako ne bi došlo do sukoba podataka. Za tu svrhu napravljena je funkcija `skini_datoteke`. Unutar funkcije se rade liste svih datoteka čiji su članovi putanje do datoteka na Dropboxu u JSON formatu. Uz standardnu autorizaciju i potrebne argumente za pristup opisane u funkciji `skini_datoteke` preko petlje `foreach` prolazi se kroz listu i datoteke se brišu sa računa te se uklanja mogućnost sinkronizacije baze sa starim podacima. Izvršavanjem ove skripte, sinkronizacija podataka je obavljena, a zaslon koji se izvršava pri pokretanju skripte vidi se na slici 14.



Slika 14: Izvođenje skripte ct_import.ps1

3.4.3. Usporedba rješenja za iste baze

U prethodna dva poglavlja predstavljena su rješenja za izvanmrežnu sinkronizaciju dviju identičnih baza podataka. Prvo od rješenja je dodavanje *export* tablica u sklopu sheme baze i izrada okidača koji se uključuju nakon umetanja, ažuriranja ili brisanja podataka. Okidači nakon nekog od prethodno navedenih događaja u *export* tablice unose promjenjene podatke i vrstu operacije. Prednosti ovakvog pristupa su:

- Postoji potpuna kontrola nad redcima u *export* tablicama
- Kreiranje tablica i datoteka na način koji najviše odgovara trenutnim potrebama
- Može se vidjeti cijela povijest promjena nad podacima

Neke od mana pristupa su:

- Dodavanje dodatnih tablica u shemu baze
- Potrebno je deaktivirati okidače kod svakog uvoza podataka
- Moguće je gomilanje podataka u dodatnim tablicama ukoliko se ne postavi brisanje podataka

Drugi način sinkronizacije je Change Tracking, rješenje u sklopu SQL Servera koji pomaže u praćenju promjena nad tablicama. Nakon omogućavanja na nivou baze i na nivou svake tablice sve je spremno za rad. Rješenje sprema tip operacije i primarni ključ redka koji je mijenjan te se uz dodatne SQL operatore podaci spajaju sa onima iz izvornih tablica. Na taj način se u transfer datotekama nalaze svi potrebni podaci. Prednosti ovakvog pristupa su:

- Jednostavno postavljanje
- Nije potrebno dodavati dodatne tablice u shemu
- Dostupno na svim verzijama SQL Servera

Neke od mana pristupa su:

- Ne vidi se povijest promjena nego samo posljednja promjena
- Potrebno je postaviti „dobar“ period zadržavanja i čišćenja tablica
- Izvlačenje podataka iz originalnih tablica teže nego u prethodnom primjeru

Usporedbom ovih dvaju pristupa može se doći do zaključka kako i jedan i drugi mogu obavljati isti posao, uz neke iznimke koje mogu i ne moraju biti loše. Okidači i *export* tablice predstavljaju rješenje na nivou samostalnog razmišljanja o problemu i kao takvo može se podesiti da radi onako kako je to administrator baze zamislio. Change Tracking predstavlja rješenje koje dolazi u sklopu SQL Servera i ponaša se prema već određenim pravilima. Stoga se Change Tracking može vidjeti kao „brzo“ rješenje koje ima ugrađene funkcije za praćenje, a jedini problem predstavlja izvlačenje podataka. Okidači i *export* tablice se mogu gledati kao rješenje koje je teže konstruirati, ali je prilagodljivije budući da ga konstruira administrator baze. Sveukupno gledajući, samostalno rješenje odgovara korisnicima koji su dobro upućeni u rad SQL Servera i imaju dozvolu „nadogradnje“ postojeće sheme dodatnim tablicama, dok ugrađeno rješenje odogovara korisnicima koji žele iskoristiti već postojeće mogućnosti poslužitelja uz dodatnu intervenciju jezika SQL.

3.5. Rješenje zadatka za različite baze

Nakon kreiranja različitih baza na poslužiteljima pristupilo se pronalasku rješenja za sinkronizaciju podataka. Jedan od načina je taj da se podaci iz svih tablica umetnu u jednu tablicu koja ima uniformnu strukturu. Završni rad prikazuje korištenje takvog pristupa, a uz pomoć tri PS skripte i SQL logike izvršava se sinkronizacija podataka. Logika koja stoji iza rješenja je takva da se unutar uniformne tablice umetne naziv tablice iz koje redak dolazi sa svim podacima tog redka. Nakon toga se podaci prebace u datoteke (CSV, XML, JSON) i tablica se isprazni. Na drugom poslužitelju podaci se iz datoteka stave unutar transfer tablice i iz nje se obavlja sinkronizacija. Također, po završetku sadržaj tablice se izbriše kako ne bi zauzimao prostor. Kôd korišten za izradu uniformne tablice prikazan je u poglavlju kreiranja različitih bâza unutar skripte *napravi_tablice.sql*, a model podataka vidljiv je u tablici 26.

Tablica 26: Tablica Podaci_transfer

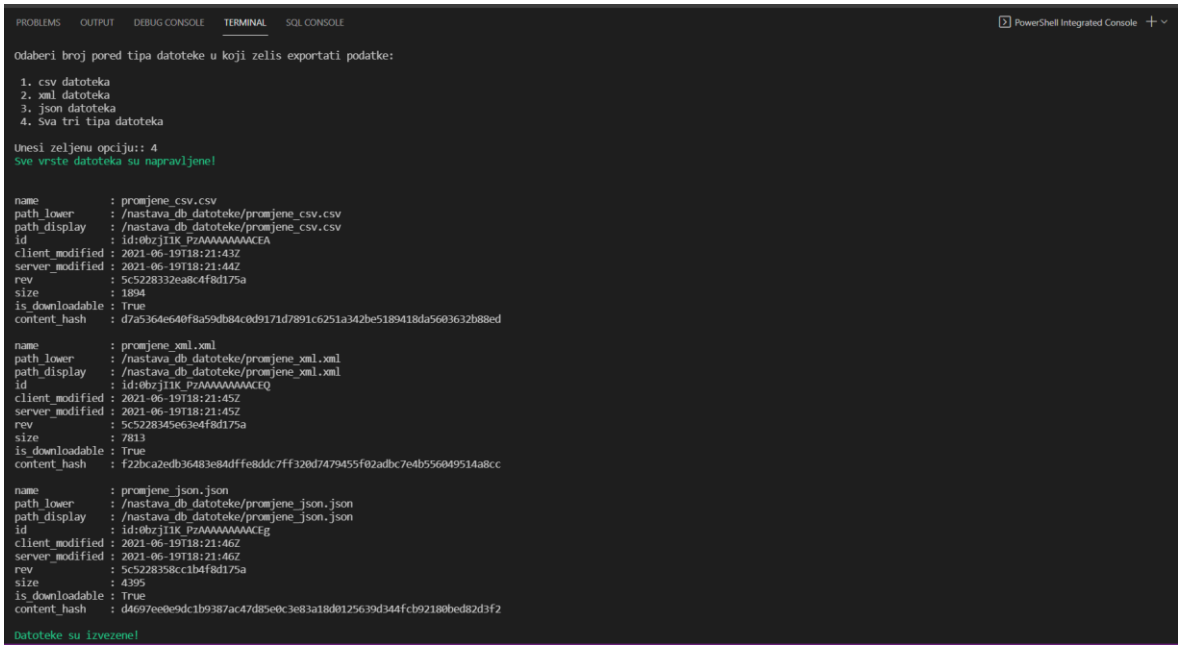
Podaci_transfer			
Naziv	**Tip**	**Veličina**	**Kardinalitet**
(PK) Podaci_transfer_ID	int	/	(1,1)
Naziv_tablice	nvarchar	32	(1,1)
Vrijednosti	nvarchar	1000	(1,1)

Tablica se sastoji od tri polja od koje prvo predstavlja primarni ključ, polje Naziv_tablice se popunjava nazivom tablice iz koje se izvlače podaci, a unutar polja Vrijednosti se umeću svi podaci iz pojedinog redka tablice. Na prvom posužitelju prvo je napravljena skripta *nastava_db_export.ps1* čiji je kôd prikazan u ispisu 24.

```
function provjeri_mape {
    $putanja_datoteke = "E:\Zavrsni_ispit\nastava_db_datoteke"
    if (!(Test-Path $putanja_datoteke)) {
        mkdir $putanja_datoteke }
    else {
        foreach ($file in Get-ChildItem $putanja_datoteke) {
            Remove-Item "E:\Zavrsni_ispit\nastava_db_datoteke\$file" }}}
function popuni_tablicu {
...
Invoke-Sqlcmd -ServerInstance $server -Database $db -Query "INSERT INTO
Podaci_transfer SELECT 'Profesor_predmet',
    CONCAT_WS('^', Profesor_pp_ID, Predmet_pp_ID) from Profesor_predmet;"
...
function upload_dropbox {
    Param([string]$SourceFilePath,              [string]$TargetFilePath,
[string]$AccessToken )
Invoke-RestMethod  -Uri  https://content.dropboxapi.com/2/files/upload  -
Method Post -InFile $SourceFilePath -Headers $headers
...
function izvezi_datoteke { param ( [int]$opt )
$access_token                                                          =
"bve5clQAe4sAAAAAAAAAAZjOMSuAugUUmCdvNILy9WxjYM8JSlOBSo_SOjRkNbAJ"
    $promjene = Invoke-Sqlcmd -ServerInstance $server -Database $db -Query
"SELECT * FROM Podaci_transfer"
    if ($opt -eq 1) {
        $promjene     |     Export-Csv     -NoTypeInformation     -Path
"E:\Zavrsni_ispit\nastava_db_datoteke\promjene_csv.csv" -Encoding UTF8
        upload_dropbox                                    -SourceFilePath
"E:\Zavrsni_ispit\nastava_db_datoteke\promjene_csv.csv" `
            -TargetFilePath   "/nastava_db_datoteke/promjene_csv.csv"   -
AccessToken $access_token
Write-Host   -ForegroundColor   Green   "Datoteka   promjene_csv.csv   je
napravljena!" }
...
function main {
    provjeri_mape
    popuni_tablicu
...
    Invoke-Sqlcmd -ServerInstance $server -Database $db -Query "TRUNCATE
TABLE Podaci_transfer" }
```

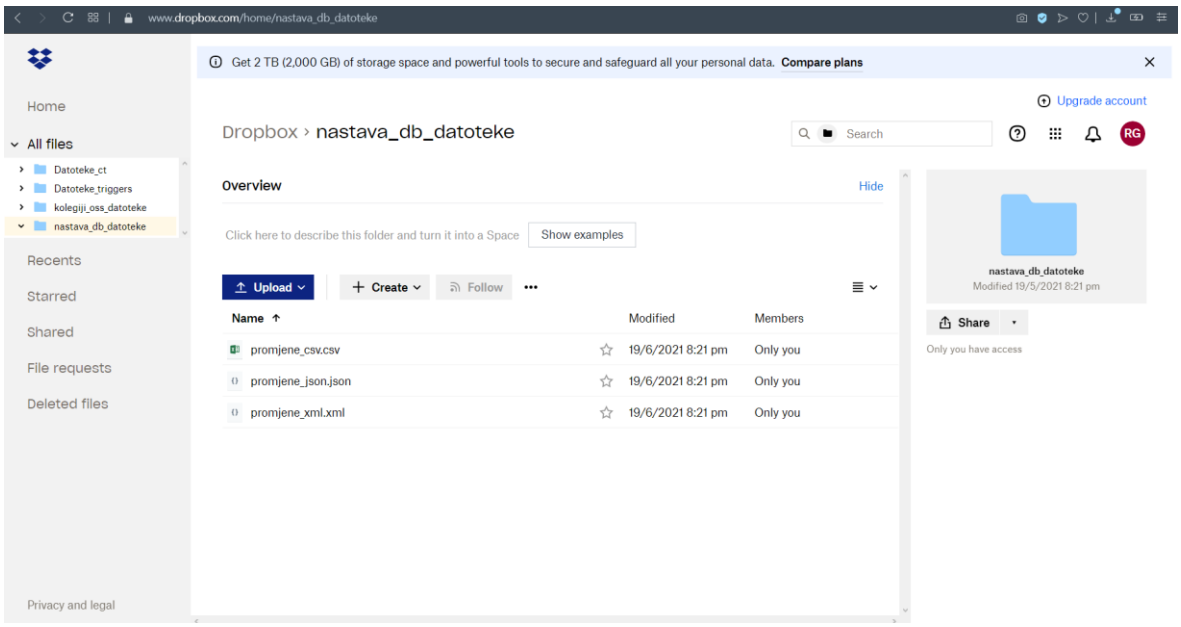
Ispis 24: Dio skripte nastava_db_export.ps1

Funkcija `main` služi za pozivanje napisanih funkcija kao i za ispis mogućnosti izvoza. Također, služi za unos odabira korisnika i za brisanje podataka iz transfer tablica. Brisanje podataka se vrši uz pomoć `TRUNCATE TABLE ime_tablice` naredbe, a za ispis se koristi `Write-Host`. Prvo se poziva funkcija `provjeri_mape`. Unutar varijable `$putanja_datoteke` postavlja se putanja do mape na računalu gdje se smještaju datoteke. Ukoliko mapa ne postoji, kreira se pomoću naredbe `mkdir $putanja_datoteke`. Ako se provjerom utvrdi da mapa postoji, sa `Remove-Item` se uklanjaju stare datoteke kako ne bi došlo do sukoba podataka. Unutar funkcije `popuni_tablicu` uz pomoć naredbe `Invoke-Sqlcmd` u transfer tablicu unose se podaci. Naredba za unos je `INSERT INTO` dok se za uniformno stavljanje svih podataka unutar polja `Vrijednosti` koristila SQL funkcija `CONCAT_WS`. `CONCAT_WS` za prvi parametar prima razdjelnik (engl. *separator*); u prikazanoj skripti to je znak „^“ sa kojim se podaci odvajaju, a nakon toga idu polja iz tablice koja se stavljaju u pojedini redak. Funkcijom `izvezi_datoteke` se od podataka iz transfer tablice stvaraju CSV, XML i/ili JSON datoteke. Kao parametar prima broj opcije koji korisnik odabere i na temelju broja izvozi datoteke u određeni format ili sve formate datoteka. Kreiranjem datoteka ispisuje se poruka o uspješnosti, a ako je odabrana kriva opcija ispisuje se poruka da se odabere druga opcija. Unutar svakog valjanog uvjeta pozivaju se funkcije `Export-Csv`, `Export-Clixml` i/ili `ConvertTo-Json` za izvoz u željenu vrstu datoteke. Nakon toga poziva se funkcija `upload_dropbox` koja postavlja datoteke na Dropbox račun. `Upload_dropbox` za argumente prima izvorišnu putanju datoteke na računalu, odredišnu putanju datoteke na Dropboxu i `Access Token`. Unutar zaglavlja se stavljaju parametri za autorizaciju i komunikaciju sa Dropbox API-em, a uz pomoć funkcije `Invoke-RestMethod` datoteke se postavljaju na Dropbox. `Invoke-RestMethod` prima parametre `-Uri` (definira internetski resurs kojem funkcija pristupa), `-Method` (HTTP metoda za rad sa podacima na internetu), `-Infile` (određuje datoteku koju prenosimo) i `-Headers` (za zaglavlje) i nakon toga datoteke su postavljene na Dropbox. Zaslon koji se prikazuje pri izvršavanju skripte prikazan je na slici 15.



Slika 15: Izvođenje skripte nastava_db_export.ps1

Postavljene datoteke nalaze se na Dropboxu i mogu se vidjeti na slici 16.



Slika 16: Datoteke na Dropboxu

Na svakom poslužitelju napravljene su skripte za izvoz koje su po logici izvođenja iste. Kako se ne bi ugrozio redoslijed skripti, objasnit će se njihovo izvođenje na jednom pa na drugom poslužitelju. Prilikom kreiranja datoteka na drugom poslužitelju može se krenuti

sa uvozom datoteka. To je ostvareno uz pomoć skripte *import_podataka.ps1* čiji se kôd vidi u ispisu 25.

```
$ErrorActionPreference = "Stop"

$server = "DESKTOP-17PP6LU\SQLEXPRESS"

$db = "nastava_db"

$Podaci_transfer = Invoke-Sqlcmd -ServerInstance $server -Database $db -
Query "Select Naziv_tablice, Vrijednosti from Podaci_transfer;"

$niz_id_student                          =                          New-Object
System.Collections.Generic.List[System.Object]

...

foreach ($item in $Podaci_transfer) {

    if ($item.Naziv_tablice -eq 'Student') {

...

elseif ($item.Naziv_tablice -eq 'Zaposlenik_kolegij') {

        $att = $item.Vrijednosti.split('^').Trim()

        $profesor_pp_ID = $att[0]

        $predmet_pp_ID = $att[1]

        $profesor_predmet  =  [System.Tuple]::Create([int]$profesor_pp_ID,
[int]$predmet_pp_ID)

        $niz_id_profesor_predmet.Add($profesor_predmet)

        $nadi = Invoke-Sqlcmd -ServerInstance $server -Database $db -Query
"SELECT 1 FROM Profesor_predmet WHERE

        Profesor_pp_ID   =   CAST('$($profesor_pp_ID)'   AS   INT)   AND
Predmet_pp_ID = CAST('$($predmet_pp_ID)' AS INT)"

        if (!($nadi)) {

            Invoke-Sqlcmd  -ServerInstance  $server  -Database  $db  -Query
"INSERT INTO Profesor_predmet VALUES(CAST('$($profesor_pp_ID)' AS INT),

            CAST('$($predmet_pp_ID)' AS INT), 'N/A')" } }

...

$idovi_profesor_predmet = Invoke-Sqlcmd -ServerInstance $server -Database
$db -Query "SELECT Profesor_pp_ID, Predmet_pp_ID FROM Profesor_predmet"

$idovi_profesor_predmet | ForEach-Object {

    $tuple_profesor_predmet  =  [System.Tuple]::Create($_.Profesor_pp_ID,
$_.Predmet_pp_ID)

    if ($tuple_profesor_predmet -notin $niz_id_profesor_predmet) {

        Invoke-Sqlcmd -ServerInstance $server -Database $db -Query "DELETE
FROM Profesor_predmet WHERE

        Profesor_pp_ID = CAST('$($tuple_profesor_predmet.Item1)' AS INT)
AND Predmet_pp_ID = CAST('$($tuple_profesor_predmet.Item2)' AS INT)" } }

...
```

Ispis 25: Dio skripte import_podataka.ps1

Postavljanjem `$ErrorActionPreference = Stop` otklanja se mogućnost preranog prekida izvođenja skripte. `$Server` je naziv poslužitelja, a `$db` naziv baze podataka na koju se skripta spaja. Prvo se unutar `$Podaci_transfer` varijable spremaju sve vrijednosti iz transfer tablice. To se obavlja preko funkcije `Invoke-Sqlcmd` i `SELECT` SQL naredbe. Također, preko funkcije `New-Object` stvaraju se liste identifikatora svake tablice koje će biti upotrijebljene kod brisanja. Petljom `Foreach` se prolazi preko objekata varijable `$Podaci_transfer`. Unutar petlje se nalaze uvjeti `if`, `elseif` i `else` koji na temelju polja `Naziv_tablice` „odlučuju“ o tome koju tablicu treba provjeriti. Nakon što je tablica određena unutar varijable `$att` spremaju se podaci iz polja `Vrijednosti` koji se odvajaju `Split` funkcijom unutar PS-a. `Split` odvaja podatke preko unesenog znaka (u prikazanom slučaju to je „^“), a podacima se može pristupiti preko indeksa. Zatim se unutar varijable `$nadi` pokušava pronaći red u tablici koji ima istu vrijednost primarnog ključa. Ako takav red ne postoji ide se na umetanje novog redka u tablicu koje se obavlja preko `Invoke-Sqlcmd` funkcije i naredbe `INSERT INTO ime_tablice VALUES`. SQL funkcijom `CAST` svi podaci su pretvoreni u tip podatka koji odgovara bazi. Moguće je da redak sa istim primarnim ključem ipak postoji pa se unutar varijable `$azuriraj` pokuša pronaći redak sa svim identičnim podacima. U slučaju da takav red ne postoji ide se na ažuriranje redka uz pomoć funkcije `Invoke-Sqlcmd` i naredbe `UPDATE SET`. Također se koristi funkcija `CAST` za pretvorbu podataka.

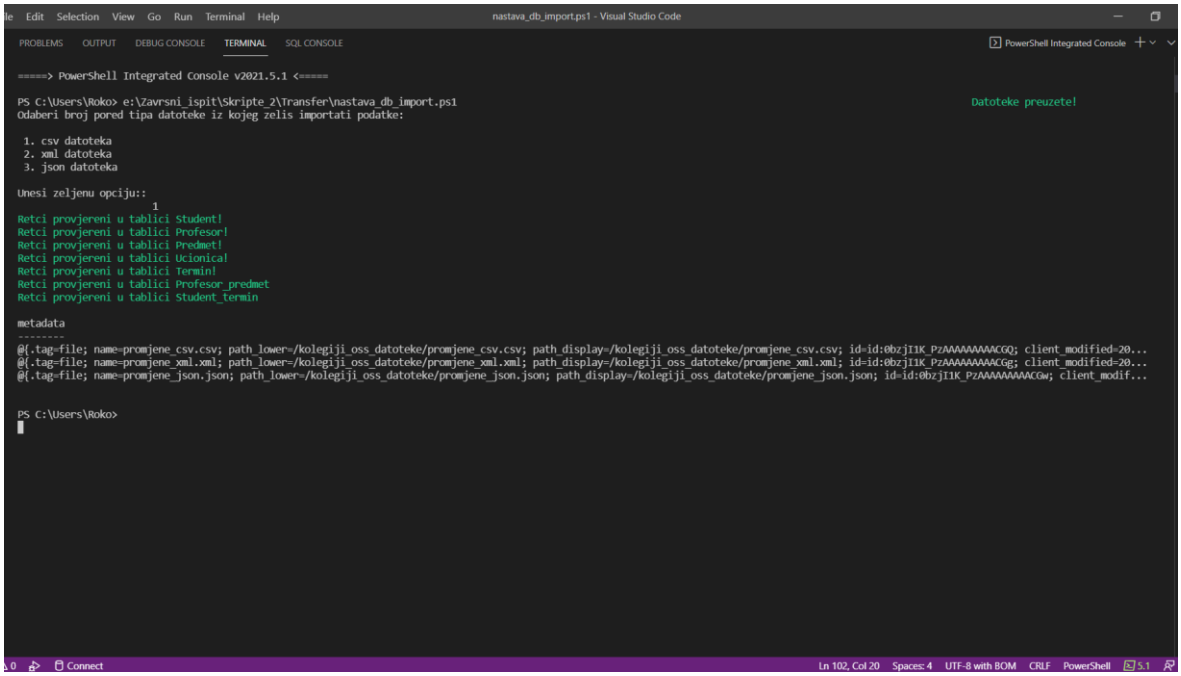
Kako su na poslužiteljima baze različitih struktura bilo je potrebno rješiti problem razlike u tablicama. Tako je na primjer u bazi *kolegiji_oss* napravljena tablica Nastava koja se u bazi *nastava_db* dijeli na tablice Termin i Ucionica. Rješenje tog problema je da se izvuče broj učionice iz tablice Nastava i pokuša ga se pronaći unutar tablice Ucionica. Ukoliko taj broj ne postoji ide se sa umetanjem novog redka u tu tablicu, a ako postoji slijedi provjera sa svim podacima i moguće ažuriranje. Što se tiče brisanja podataka iz tablica, na scenu stupaju liste identifikatora tablica sa početka skripte. Prvo se unutar varijabli `$id_ovi_imeTablice` spremaju svi identifikatori pojedinih tablica. Preko petlje `ForEach-Object` se prolazi preko svakog od identifikatora i ukoliko se ne nalazi u listi uz pomoć funkcije `Invoke-Sqlcmd` i naredbe `DELETE` redak se briše iz tablice. Na kraju skripte ispisuje se poruka o prolasku kroz tablice te se preko naredbe `TRUNCATE TABLE Podaci_transfer` podaci brišu iz tablice kako ne bi zauzimali prostor, a baze su sinkronizirane. Korisniku se na izbor daje da odluči iz koje vrste datoteke želi obaviti uvoz podataka pa je u tu svrhu napravljena skripta *nastava_db_import.ps1* čiji se kôd može vidjeti u ispisu 26.

```
$server = "DESKTOP-17PP6LU\SQLEXPRESS"
$db = "nastava_db"
function import_datoteke {
    param ([int] $opt)
    if ($opt -eq 1) { ...
elseif ($opt -eq 2) {
        $xml_podaci = Import-Clixml -Path
"E:\Zavrsni_ispit\kolegiji_oss_datoteke\promjene_xml.xml"
        $xml_podaci | ForEach-Object {
            Invoke-Sqlcmd -ServerInstance $server -Database $db -Query
"Insert into Podaci_transfer
            values(CAST(N'$($_.Naziv_tablice)' as nvarchar(32)),
CAST('$($_.Vrijednosti)' as nvarchar(1000)))" }
        .("E:\Zavrsni_ispit\Skripte_2\Transfer\import_podataka.ps1") }
...
function skini_datoteke {
    [CmdletBinding()]
    Param([string]$DropboxFolder, [string]$OutputFolder ) ...
Invoke-RestMethod -Method POST -Uri
"https://content.dropboxapi.com/2/files/download_zip" -Headers $headers
-OutFile $Tempfolder -ContentType ""
...
function ukloni_datoteke { ...
foreach ($i in $lista_datoteke) {
        try {
            Invoke-RestMethod -Method POST -Uri
"https://api.dropboxapi.com/2/files/delete_v2"
            -Headers $headers -Body $i } } ...
function main {
    try {
        $mapa_dropbox = "/kolegiji_oss_datoteke"
        $mapa_pc = "E:\Zavrsni_ispit"
        skini_datoteke -DropboxFolder $mapa_dropbox -OutputFolder
$mapa_pc
        Write-Host -ForegroundColor Green "Datoteke preuzete!" } ...
```

Ispis 26: Dio skripte nastava_db_import.ps1

Funkcija `main` se koristi za preuzimanje datoteka, ispis mogućnosti uvoza, unos opcije korisnika te za poziv ostalih funkcija. Unutar `try` iznimke se poziva funkcija `skini_datoteke` koja preuzima datoteke sa Dropboxa, a ako `try` ne uspije `catch` iznimka

ispisuje grešku. `Skini_datoteke` za argumente prima izvorišnu mapu na Dropboxu i mapu na računalu gdje se spremaju preuzeti podaci. Također se dodaju parametri za autorizaciju i komunikaciju sa Dropbox API-em. Varijabla `$Tempfolder` predstavlja privremenu mapu za spremanje na računalu budući da se podaci nalaze u *zip* formatu. Nakon toga se uz pomoć funkcije `Invoke-RestMethod` datoteke preuzimaju sa Dropboxa. `Invoke-RestMethod` prima parametre `-Uri` (definira internetski resurs kojem funkcija pristupa), `-Method` (HTTP metoda za rad sa podacima na internetu), `-Infile` (određuje datoteku koju preuzimamo) i `-Headers` (za zaglavlje). `Expand-Archive` služi za otpakiravanje datoteke sa `$Tempfolder` lokacije na željenu lokaciju na računalu, a `Remove-Item` se koristi za uklanjanje nepotrebnih datoteka. `Import_datoteke` na temelju unosa korisnika pokreće funkcije `Import-Csv`, `Import-Clixml` ili `ConvertFrom-Json` koje uvoze podatke iz datoteka u varijable. Petljom `ForEach-Object` se prolazi kroz objekte pojedinih varijabli i umeću se u `Podaci_transfer` tablicu preko funkcije `Invoke-Sqlcmd` i `INSERT INTO ime_tablice VALUES` naredbe. Nakon toga se navodi putanja do skripte *import_podataka.ps1* koja obavlja uvoz podataka. Nakon uvoza podataka potrebno je ukloniti datoteke sa Dropboxa kako ne bi došlo do sukoba podataka. Za tu svrhu napravljena je funkcija `skini_datoteke`. `Skini_datoteke` stvara listu putanja do Dropbox datoteka koje je potrebno ukloniti u JSON formatu. Petljom `Foreach` prolazi se kroz listu i preko `try` iznimke se pokušavaju ukloniti datoteke koristeći funkciju `Invoke-RestMethod`, a ako se neke datoteke ne nalaze na Dropboxu unutar `catch` iznimke se ispisuje upozorenje. Ovom skriptom omogućen je odabir načina sinkronizacije podataka, a zaslon koji se prikazuje pri izvođenju skripte prikazan je na slici 17.



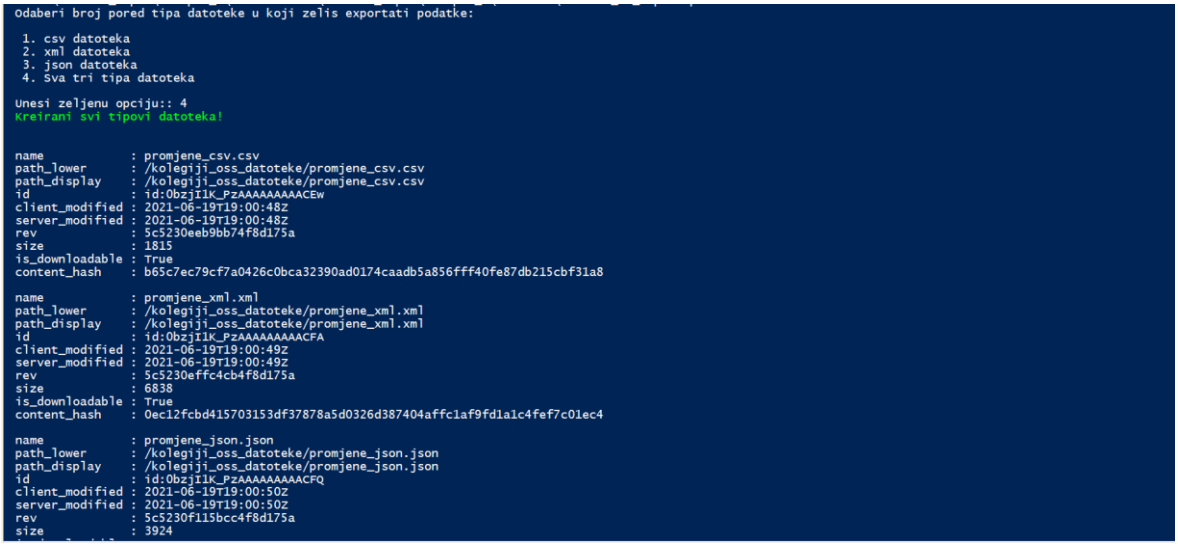
Slika 17: Izvođenje skripte nastava_db_import.ps1

Što se tiče poslužitelja na kojem se nalazi baza kolegiji_oss, skripta *kolegiji_oss_export.ps1* prikazana je u ispisu 27.

```
function napravi_mape {
    $putanja_datoteke = "C:\Zavrsni_ispit\kolegiji_oss_datoteke"
    if (!(Test-Path $putanja_datoteke)) {
        mkdir $putanja_datoteke }
    else {
        foreach ($file in Get-ChildItem $putanja_datoteke) {
            Remove-Item "C:\Zavrsni_ispit\kolegiji_oss_datoteke\$file"
      } } }
...
function popuni_tablicu {
Invoke-Sqlcmd -ServerInstance $server -Database $db -Query "INSERT INTO
Podaci_transfer SELECT 'Student_nastava',
    CONCAT_WS('^', Student_sn_ID, Nastava_sn_ID) from Student_nastava;"
function upload_dropbox {
    Param([string]$SourceFilePath,               [string]$TargetFilePath,
[string]$AccessToken )
Invoke-RestMethod  -Uri  https://content.dropboxapi.com/2/files/upload  -
Method Post -InFile $SourceFilePath -Headers $headers }
...
function izvezi_datoteke {
    param ([int]$opt)
    $access_token                                                          =
"bve5clQAe4sAAAAAAAAAAZjOMSuAugUUmCdvNILy9WxjYM8JSlOBSo_SOjRkNbAJ"
    $promjene = Invoke-Sqlcmd -ServerInstance $server -Database $db -Query
"SELECT * FROM Podaci_transfer"
    if ($opt -eq 1) {
        $promjene      |      Export-Csv      -NoTypeInformation      -Path
"C:\Zavrsni_ispit\kolegiji_oss_datoteke\promjene_csv.csv" -Encoding UTF8
        upload_dropbox                                          -SourceFilePath
"C:\Zavrsni_ispit\kolegiji_oss_datoteke\promjene_csv.csv" `
            -TargetFilePath  "/kolegiji_oss_datoteke/promjene_csv.csv"  -
AccessToken $access_token
        Write-Host -ForegroundColor Green "Kreirana CSV datoteka!"}
...
function main {
    napravi_mape
    popuni_tablicu
Invoke-Sqlcmd -ServerInstance $server -Database $db -Query "TRUNCATE TABLE
Podaci_transfer" }
...
```

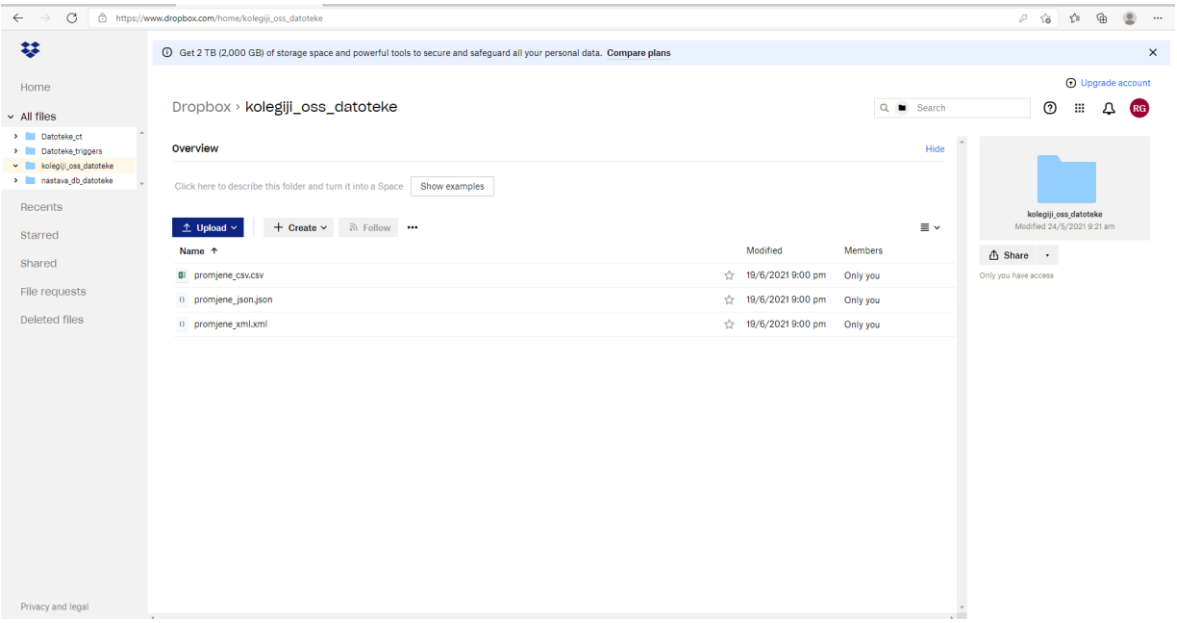
Ispis 27: Dio skripte kolegiji_oss_export.ps1

Zaslon koji se pojavljuje prilikom izvođenja skripte prikazan je na slici 18.



Slika 18: Izvršavanje skripte kolegiji_oss_export.ps1

Datoteke su postavljene na Dropbox i vide se na slici 19.



Slika 19: Postavljene datoteke na Dropboxu

Skripta *import_podataka.ps1* sa drugog poslužitelja prikazana je u ispisu 28.

```
$ErrorActionPreference = "Stop"

$server = "DESKTOP-D4UT8J3\SQLEXPRESS"

$db = "kolegiji_oss"

$Podaci_transfer = Invoke-Sqlcmd -ServerInstance $server -Database $db -
Query "Select Naziv_tablice, Vrijednosti from Podaci_transfer;" ...

$niz_id_zaposlenik                          =                          New-Object
System.Collections.Generic.List[System.Object]

...

foreach ($item in $Podaci_transfer) {

    if ($item.Naziv_tablice -eq 'Student') {...

elseif ($item.Naziv_tablice -eq 'Profesor_predmet') {

        $att = $item.Vrijednosti.split('^').Trim()

        $zaposlenik_zk_ID = $att[0]

        $kolegij_zk_ID = $att[1]

        $zaposlenik_kolegij                                                  =
[System.Tuple]::Create([int]$zaposlenik_zk_ID, [int]$kolegij_zk_ID)

        $niz_id_zaposlenik_kolegij.Add($zaposlenik_kolegij)

        $nadi = Invoke-Sqlcmd -ServerInstance $server -Database $db -Query
"SELECT 1 FROM Zaposlenik_kolegij WHERE

        Zaposlenik_zk_ID  =  CAST('$($zaposlenik_zk_ID)'  AS  INT)  AND
Kolegij_zk_ID = CAST('$($kolegij_zk_ID)' AS INT)"

        if (!($nadi)) {

            Invoke-Sqlcmd -ServerInstance $server -Database $db -Query
"INSERT  INTO  Zaposlenik_kolegij  VALUES(CAST('$($zaposlenik_zk_ID)'  AS
INT),

            CAST('$($kolegij_zk_ID)' AS INT))" } }

...

$idovi_zaposlenik_kolegij  =  Invoke-Sqlcmd  -ServerInstance  $server  -
Database  $db  -Query  "SELECT  Zaposlenik_zk_ID,  Kolegij_zk_ID  FROM
Zaposlenik_kolegij"

$idovi_zaposlenik_kolegij | ForEach-Object {

    $tuple_zaposlenik_kolegij                                                =
[System.Tuple]::Create($_.Zaposlenik_zk_ID, $_.Kolegij_zk_ID)

    if ($tuple_zaposlenik_kolegij -notin $niz_id_zaposlenik_kolegij) {

        Invoke-Sqlcmd -ServerInstance $server -Database $db -Query "DELETE
FROM Zaposlenik_kolegij WHERE

        Zaposlenik_zk_ID  =  CAST('$($tuple_zaposlenik_kolegij.Item1)'  AS
INT)  AND  Kolegij_zk_ID  =  CAST('$($tuple_zaposlenik_kolegij.Item2)'  AS
INT)" } }

...
```

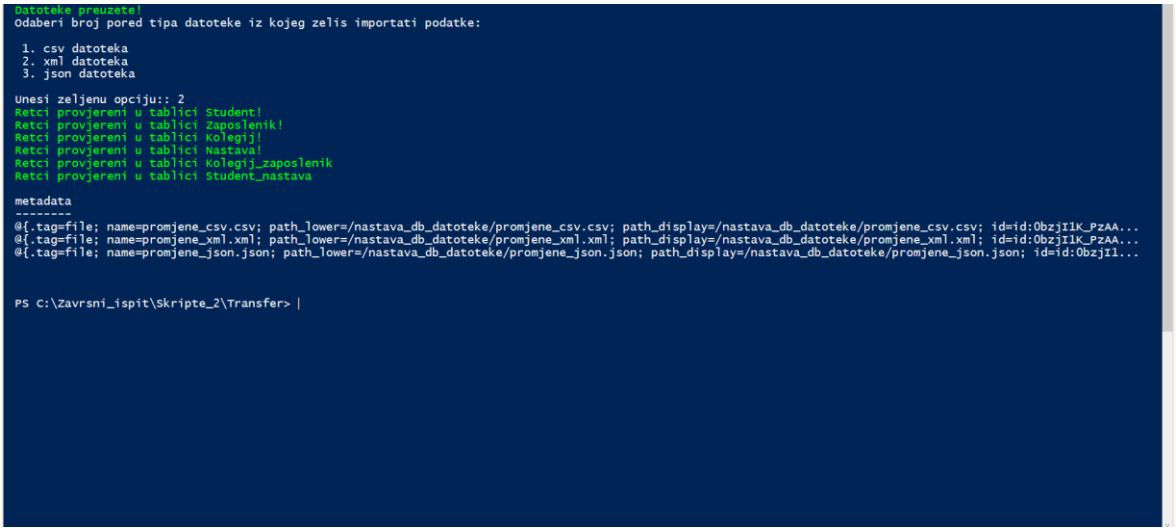
Ispis 28: Dio skripte import_podataka.ps1

Skripta *kolegiji_oss_import.ps1* prikazana je u ispisu 29.

```
$server = "DESKTOP-D4UT8J3\SQLEXPRESS"
$db = "kolegiji_oss"
function import_datoteke {
    param ([int] $opt)
    if ($opt -eq 1) {
        $csv_podaci = Import-Csv -Path "C:\Zavrsni_ispit\nastava_db_datoteke\promjene_csv.csv"
        $csv_podaci | ForEach-Object {
            Invoke-Sqlcmd -ServerInstance $server -Database $db -Query "Insert into Podaci_transfer
            values(CAST('$($_.Naziv_tablice)' as nvarchar(32)), CAST('$($_.Vrijednosti)' as nvarchar(1000)))" }
        .("C:\Zavrsni_ispit\Skripte_2\Transfer\import_podataka.ps1") }
function skini_datoteke {
    [CmdletBinding()]
    Param([string]$DropboxFolder, [string]$OutputFolder)
Invoke-RestMethod –Method POST `
        -Uri "https://content.dropboxapi.com/2/files/download_zip" `
        –Headers $headers –OutFile $Tempfolder –ContentType "" }
...
function ukloni_datoteke {
foreach ($i in $lista_datoteke) {
        try {
            Invoke-RestMethod –Method POST –Uri "https://api.dropboxapi.com/2/files/delete_v2"
            –Headers $headers -Body $i } }
...
function main {
    try {
        $mapa_dropbox = "/nastava_db_datoteke"
        $mapa_pc = "C:\Zavrsni_ispit"
        skini_datoteke -DropboxFolder $mapa_dropbox -OutputFolder $mapa_pc
        Write-Host -ForegroundColor Green "Datoteke preuzete!" }
...
```

Ispis 29: Dio skripte kolegiji_oss_import.ps1

Zaslon koji se prikazuje pri izvršavanju skripte *kolegiji_oss_import.ps1* vidljiv je na slici 20.



Slika 20: Izvođenje skripte kolegiji_oss_import.ps1

Logika predstavljenih skripti je identična onoj kod baze *nastava_db*. Razlikuju se u putanjama do datoteka i nazivima SQL poslužitelja i baze te malo drukčijem načinu unosa podataka u tablice (različito oblikovani podaci i tipovi). Skripte i zasloni su stavljeni zbog toga što se prikazuje rješenje za baze različite strukture. Izvođenjem predstavljenih skripti na bilo kojem od dva poslužitelja dobivaju se identični podaci i problem različitosti je riješen.

4. ZAKLJUČAK

Cilj završnog rada bila je izrada skripti za izvanmrežnu sinkronizacija SQL bâza podataka koja je ostvarena uz pomoć jezika SQL za upravljanje bazama i skriptnog jezika PS.

Ova tema je odabrana zbog sve bržeg razvoja digitalnih spremišta podataka koja su gotovo u potpunosti zamijenila ona fizička. Sve više i više podataka sprema se u baze podataka, a aplikacije koje se izrađuju, bilo za računala ili za pametne telefone, koriste te podatke. Jedan od glavnih uvjeta svake aplikacije su informacije koje se prezentiraju krajnjim korisnicima. Također, aplikacije koje služe za istu svrhu, a namjenjene su različitim osobama mogu pristupati bazama na različitim lokacijama. Isto tako, mogu imati i različitu strukturu. Takve baze moraju biti redovno sinkronizirane kako bi podaci unutar njih bili ispravni i to je krajnja svrha ovog završnog rada.

Završni rad je temeljen na dva tipa sinkronizacije podataka; podacima u istim i različitim bazama. Što se tiče istih baza, korištena su dva pristupa: okidači i *export* tablice te SQL Server Change Tracking. U prikazu rješenja temeljito je objašnjen način izrade te prednosti i mane svakog od pristupa. Ukratko, Change Tracking predstavlja već postojeće rješenje koje je potrebno aktivirati na poslužitelju, a *export* tablice i okidače potrebno je napraviti kroz skripte. Kod različitih baza korištena je samo jedna transfer tablica koja je sadržavala sve potrebne podatke. Podaci su se uz pomoć logike i funkcija unutar SQL-a i jezika PS uspješno sinkronizirali. Takav pristup je bio potreban zbog različite strukture baza na poslužiteljima, a dobro promišljen i izrađen predstavlja jedno od boljih rješenja za izvanmrežnu sinkronizaciju baza podataka.

Neka od područja primjene prikazanih rješenja (osim prethodno spomenutih) svakako se mogu naći u poslovnoj informatici. Može se zamisliti primjer gdje je tvrtka sa sjedištem u inozemstvu (npr. Češka) otvorila podružnicu u Hrvatskoj. Postoje dvije baze podataka koje služe za istu svrhu međutim nisu međusobno povezane. Tvrtka svakodnevno šalje podatke podružnici u Hrvatskoj, a onda se podaci po unaprijed određenom rasporedu uvoze u bazu (podaci poslani nakon radnog vremena, a sinkronizacija se obavi preko noći). Na ovaj način svaki radni dan započinje sa bazama podataka koje su identične i pružaju identične podatke bilo u inozemstvu ili u Hrvatskoj.

Može se zaključiti kako je predstavljeno rješenje itekako potrebno u današnjem svijetu budući da postoji na desetke slučajeva korištenja ovakvih tehnologija, a podaci su izrazito bitan segment računarstva. Predstavljeni rad nije nešto novo u svijetu računarstva budući da već postoji dosta alata koji po automatizmu sinkroniziraju baze. Međutim postoje situacije kada alati nisu dovoljni ili ne mogu obaviti dobar posao zbog specifičnosti zadatka. Tada je potrebno pronaći drugo rješenje koje zadovoljava potrebe i radi u skladu sa očekivanjima. Administrator baze podataka tada pred sobom ima zadatak, a ovaj rad prikazuje rješenje izvanmrežne sinkronizacije SQL baza podataka.

LITERATURA

[1] Robert Polding, PhD, „Databases: Evolution and Change“, https://medium.com/@rpolding/databases-evolution-and-change-29b8abe9df3e (posjećeno 31.05.2021)

[2] Torba, Z.: Baze podataka, Veleučilište u Splitu, 2001.

[3] Computer notes, „What is a Database Server“, https://ecomputernotes.com/fundamental/what-is-a-database/what-is-a-database-server

(posjećeno 01.06.2021.)

[4] Microsoft, „Editions and supported features of SQL Server 2019 (15.x)“, https://docs.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-version-15?view=sql-server-ver15 (posjećeno 01.06.2021.)

[5] Microsoft, „What is Powershell?“, https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.1#configuration-management (posjećeno 02.06.2021.)

[6] Dropbox, „What is Dropbox?“, https://www.dropbox.com/features?trigger=global_footer (posjećeno 02.06.2021.)

[7] ByteScout, „CSV format: History, advantages and why it is still popular“, https://bytescout.com/blog/csv-format-history-advantages.html#2 (posjećeno 02.06.2021.)

[8] Hubspot, „XML Files: What They Are & How To Open Them“, https://blog.hubspot.com/website/what-is-xml-file (posjećeno 02.06.2021.)

[9] BegginersBook, „Advantages and disadvantages of XML“, https://beginnersbook.com/2018/10/advantages-and-disadvantages-of-xml/ (posjećeno 02.06.2021)

[10] Martin Drapeau, „Our friends CSV and JSON“, https://medium.com/@martindrapeau/the-state-of-csv-and-json-d97d1486333 (posjećeno 03.06.2021.)

[11] Geekboots, „JSON with advantage and disadvantage“, https://www.geekboots.com/story/json-with-advantage-and-disadvantage

(posjećeno 03.06.2021.)

[12] Gravity docs, „Creating a Custom Dropbox App“, https://docs.gravityforms.com/creating-a-custom-dropbox-app/ (posjećeno 07.06.2021.)

[13] Geeksforgeeks, „SQL Trigger | Student Database“, https://www.geeksforgeeks.org/sql-trigger-student-database/ (posjećeno 07.06.2021.)

[14] Microsoft, „Microsoft.PowerShell.Utility“, https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/?view=powershell-7.1

(posjećeno 07.06.2021.)

[15] Laurent Kempé, „Upload files to DropBox from PowerShell“, https://laurentkempe.com/2016/04/07/Upload-files-to-DropBox-from-PowerShell/ (posjećeno 07.06.2021.)

[16] CloudShell, „Downloading Folders from Dropbox Using Powershell“, https://autoworld7car.wordpress.com/2020/02/23/downloading-files-from-dropbox-using-powershell/ (posjećeno 07.06.2021.)

[17] Dropbox, „Delete Folder Using PowerShell“, https://www.dropboxforum.com/t5/Dropbox-API-Support-Feedback/Delete-Folder-Using-PowerShell/td-p/302667 (posjećeno 08.06.2021.)

[18] SQLShack, „Creating a SQL Server audit using SQL Server Change Tracking“, https://www.sqlshack.com/creating-a-sql-server-audit-using-sql-server-change-tracking/ (posjećeno 08.06.2021.)