Mehanizmi sinkronizacije između baza podataka

Sirovec, Lucia

SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
Ak. god. 2020./2021.

Lucia Sirovec

Mehanizmi sinkronizacije između baza podataka

Završni rad

Mentor: dr.sc. Vedran Juričić, docent

Zagreb, srpanj 2021.

Izjava o akademskoj čestitosti

Izjavljujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Za mog tatu, koji je vjerovao u najbolje u meni i za ekipu ispred knjižnice koja je nastavila to činiti umjesto njega.

Sadržaj

1. Uvod

Razvojem tehnologije ljudi su omogućili akumulaciju velikog broja podataka. Njih je potrebno organizirati kako bi mogli biti efikasno iskorišteni. U računalnom svijetu su u tu svrhu nastale baze podataka.

Podatak (engl. *data*) je činjenica za koju se zna da se dogodila, da postoji ili da je istinita (Znanje, 2021). U računalnim znanostima potrebna je pragmatičnija definicija pa stoga podatak možemo definirati kao formalizirani znakovni prikaz činjenica, pojmova i naredaba pogodan za priopćavanje, interpretiranje te analognu i digitalnu obradu (Kiš, 2002) ili pojednostavljeno kao male cjeline koje se pohranjuju u bazama podataka (Technopedia Inc., 2021). Baza je u svojoj rječničkoj definiciji osnova ili temelj, a u računalnim znanostima i označava početak iz kojeg se počinje graditi informacijski sustav (Znanje, 2021).

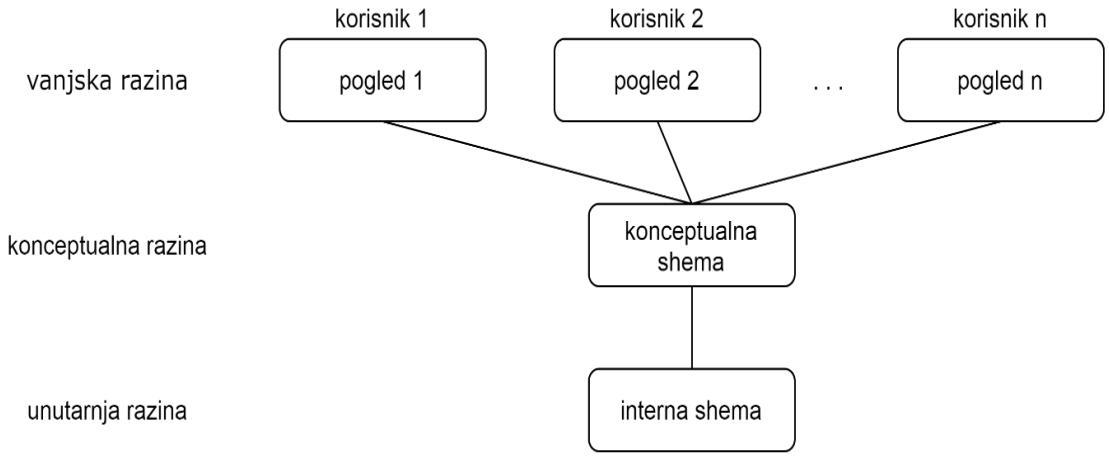
Baze podataka (engl. *database*) nemaju jedinstvenu definiciju, no njihov općeniti opis jest da su zbirka fizički pohranjenih podataka (Date, 2006). Sintagma „baza podataka“ tako može označavati cijeli niz različitih organiziranih podataka, od onih jednostavnijih, pr. telefonski imenik, do kompleksnijih, kao što je cjelokupni sustav podataka neke obrazovne ustanove (Riordan, 1999). One su nezaobilazna osnova mnogih sustava koji se danas koriste. Osmišljene su kako bi se podaci mogli međusobno povezivati i tako stavljati u red čime se znatno olakšava njihovim baratanjem i povećava korisnost podataka. U ovom radu fokus je na bazama podataka u smislu temeljnog fonda podataka i informacija koji služi u informatičkoj obradi u pojedinim poslovima i strukama (Znanje, 2021). Budući da je baza podataka kombinacija strukture i pohranjenih podataka, potrebno je smjestiti ju u sustav koji će omogućiti prikupljanje, upis, obradu i prikaz podataka. U tu svrhu osmišljeni su sustavi za upravljanje bazama podataka (engl. *database management system*) koji imaju ulogu posrednika između korisnika i podataka (Pavlić, 2011).

U današnjem globaliziranom svijetu sustavima baza podataka moguće je pristupiti i s više mjesta u isto vrijeme. Taj privilegij ipak može dovesti do problema ukoliko se podaci pristupom mijenjaju te su iz tog razloga osmišljeni mehanizmi kojima se to sprječava. Podaci u bazama podataka kao i svi fizički podaci nisu sigurni od gubitka, te je potrebno ugraditi mehanizme kojima se sprječava gubitak podataka (ili ostvaruje mogućnost njihovog povratka).

U drugom poglavlju ovog rada bit će opisani općeniti koncepti vezani uz baze podataka – modeli baza podataka, jezici za upravljanje bazama podataka, sustavi za upravljanje bazama podataka i različiti tipovi baza podataka (centralizirane, decentralizirane, standardne, mobilne). U trećem poglavlju bit će opisani mehanizmi kojima se koriste sustavi za upravljanje baza podataka kako bi međusobno sinkronizirali podatke i načini kojima se održava rad baza podataka. U četvrtom poglavlju biti ukratko prikazani i uspoređeni modeli nekih profesionalnih rješenja, a u petom sustavi koji koriste sinkronizaciju, a nisu baze podataka u užem smislu.

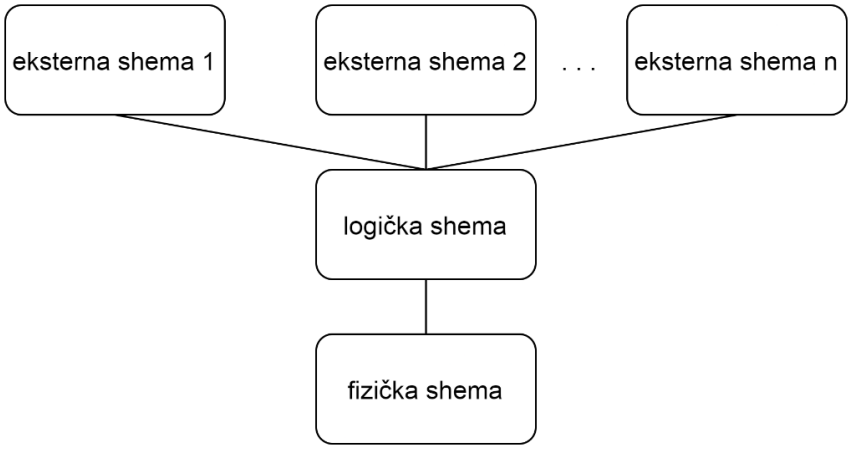
2. Baza podataka

Baze podataka opisuju se ANSI/SPARC (engl. *American National Standards Institute, Standards Planning and Requirements Committee*) arhitekturom osmišljenom 1971. godine od strane DBTG (*Data Base Task Group*). Tri razine arhitekture baze podataka su: vanjska razina, konceptualna razina i unutarnja razina. Vanjska razina je pogled na bazu. Unutarnja razina je način na koji se podaci fizički spremaju, a konceptualna razina povezuje vanjsku i unutarnju razinu te je razina kojom se bave programeri i administratori baza podataka (Ullman, 1982). Arhitekturu baza podataka zato možemo grafički prikazati kao što je prikazano na grafikonu 1.



Grafikon 1- arhitektura baze podataka (Ullman, 1982)

Svaka od razina ima i pripadajuću shemu, koje se mogu definirati kao dizajn baze podataka. Fizička (interna) shema pripada unutarnjoj razini i uključuje detalje o fizičkoj strukturi pohrane podataka i pristupa podacima. Logička (konceptualna) shema vezana je uz konceptualnu razinu i opisuje podatke u njihovim oblicima entiteta, atributa i veza te se može opisati modelima baza podataka. Eksterna shema odnosi se na dizajn baze podataka kako ju vidi korisnik, odnosno kako korisnik manipulira bazom podataka. S obzirom na različite potrebe korisnika, može postojati više eksternih shema za jednu bazu podataka, dok su za svaku bazu postoji samo jedna fizička i samo jedna logička shema (Pavlić, 2011). Kao i arhitekturu baza podataka, moguće je i njihovu shemu jednostavno prikazati kao što je prikazano u grafikonu 2.



Grafikon 2- sheme baze podataka (Varga, 2020)

2.1. Modeli baza podataka

U svakom području postoje općeprihvaćeni pojmovi i koncepti, odnosno terminologija kojom se objašnjavaju. U ovom poglavlju rada bit će objašnjeni osnovni pojmovi entiteta, atributa i veza te modeli baza podataka koji su najčešći u upotrebi od početka njihova razvoja.

Entitet (engl. *entity*) nije precizno definiran, već općenito kao pr. stvar (Date, 2006) ili jedinstveni, odvojeni objekt koji se pohranjuje u bazi podataka (Garcia-Molina, et al., 2002), nešto o čemu sustav pohranjuje informacije (Riordan, 1999). I u praksi, entitet može biti gotovo bilo što – obično neka kategorija pod koju može spadati više povezanih podataka, pr. životinja. Podaci koji bi pripadali toj kategoriji nazivaju se instance (engl. *instance*), a instance u slučaju životinja mogle bi biti pas, mačka ili konj. Problem definiranja entiteta uvijek je ovisan o potrebama korisnika, jer bi jedna od instanci, primjerice pas, drugom korisniku mogla biti entitet u kojem slučaju su instance maltezer ili ovčar (Ullman, 1982).

Atribut ili obilježje (engl. *attribute*) je karakteristika kojom se entitet može klasificirati, specificirati ili samo opisati. Atributi imaju svoje vrijednosti koje mogu biti numeričke (starost ili datum cijepljenja psa), niz znakova (pasmina), slika, itd. Važno je da je atribut jedinstven i jednoznačan entitetu u jednom trenutku vremena, a postoje promjenjivi i nepromjenjivi atributi, pr. pasmina kod pasa je nepromjenjiva, ali starost se svake godine mijenja (Ullman, 1982).

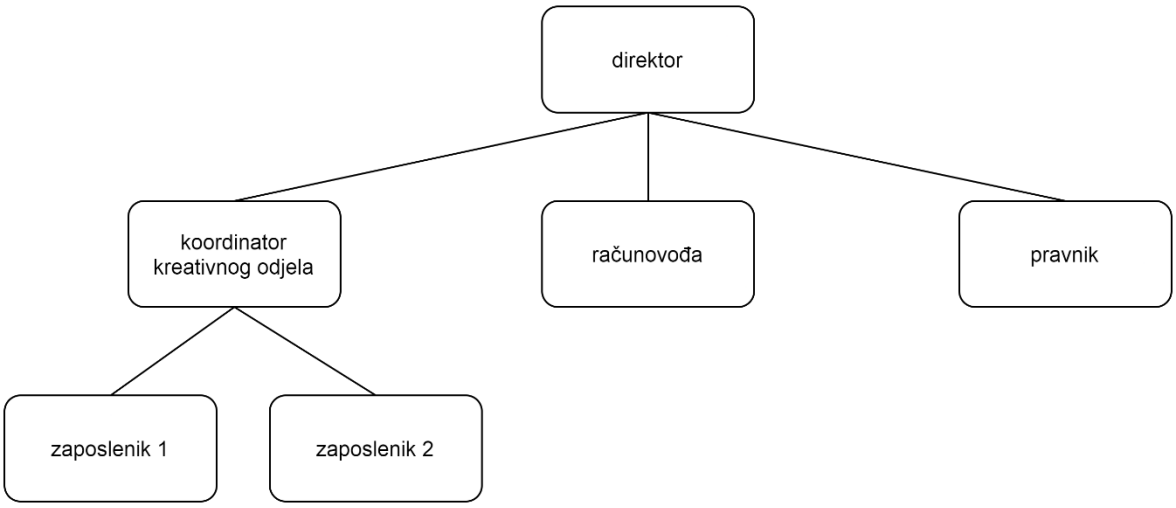
Veza (engl. *relationship*) se odnosi na vezu između entiteta u bazi podataka (Ullman, 1982). S obzirom na broj entiteta u vezi, postoji više tipova: jedan-naprama-jedan (jedan entitet je povezan samo s drugim entitetom), jedan-naprama-više (jedan entitet ima više povezanih

entiteta), više-naprama-jedan (više entiteta povezano je s jednim) i više-naprama-više (više entiteta su povezani jedni s drugima) (Varga, 2020).

Tijekom razvoja baza podataka stvarale su se razne logičke strukture koje nazivamo modelima, a tri glavna modela su: hijerarhijski, mrežni i relacijski. Model baze podataka (engl. *data model*) je jednostavan prikaz sustava, odnosno entiteta, atributa i veza između njih. Svaki model sastoji se od tri dijela, a to su strukture, ograničenja i operatori. Strukture modela podataka (engl. *structure*) grupiraju podatke koji će se nalaziti u sustavu baza podataka i tako pojmove koji se koriste u poslovanju pretvaraju u entitete, veze i atribute. Ograničenja modela podataka (engl. *constraints*) u sustavu baza podataka limitiraju svojstva entiteta i atributa kako bi odgovarali stvarnosti, definiraju dopuštene vrijednosti podataka i povezivanje između podataka. Operatori modela podataka (engl. *operators*) su koncepti koji omogućuju izmjene podataka u sustavu u skladu s promjenama u stvarnosti (Pavlić, 2011).

2.1.1. Hijerarhijski model

Hijerarhijski model (engl. *hierarchical data model*) koristi samo veze jedan-naprama-jedan i jedan-naprama-više između entiteta i to u strukturi obrnutog stabla, tj. odnosa „roditelj“ - „dijete“. Ime modela dolazi od stupnjevanja ovlaštenja koje se naziva hijerarhija (Znanje, 2021), a osmišljen je 60-ih u suradnji North American Rockwell i IBM-a (IBM, 2021). Iako prikladan za neke vrste podataka, hijerarhijski model nije prilagodljiv kompleksnijim strukturama već više odgovara sustavima kojima je fokus spremiti u bazu podataka jedan postojeći hijerarhijski sustav (Ullman, 1982). Uzmimo primjerice jedno malo poslovno okruženje, tvrtku koja se bavi stvaranjem reklama. Postoji direktor, odnosno glavna odgovorna osoba kojoj svi ostali polažu račune. Imamo pravnika, računovođu i par zaposlenika kreativnog dijela posla koji imaju svog koordinatora. Ta hijerarhija vidljiva je na primjeru grafikona 3.

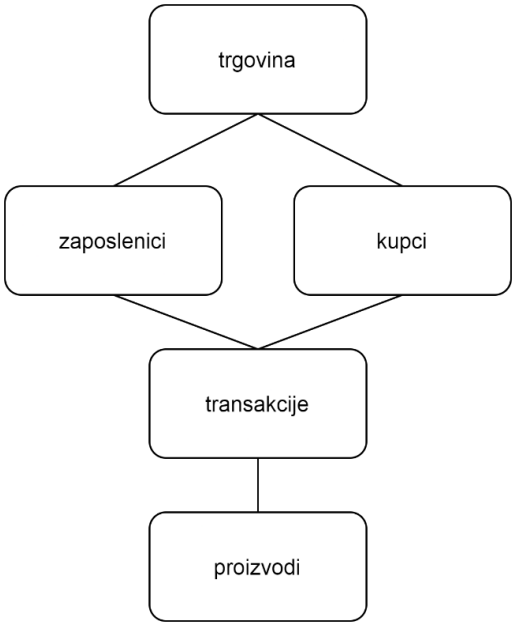


Grafikon 3- primjer hijerarhijske strukture

O svim zaposlenicima postoje podaci i svi svojim radom stvaraju podatke koje je potrebno isto tako sačuvati. Za svakog zaposlenog tako bi se mogao stvoriti posebni entitet s njegovim podacima ili stvaralaštvom. No, važno je primijetiti da se u hijerarhijskom modelu više entiteta povezuje samo s „roditeljskim“ entitetom (koordinator, računovođa i pravnik su povezani samo s direktorom), dok ti „roditeljski“, odnosno korijenski entiteti mogu imati više „djece“ (koordinator je korijenski entitet dvama zaposlenicima).

2.1.2. Mrežni model

Mrežni model (engl. *network data model*) kao i hijerarhijski koristi veze jedan-naprama-jedan i jedan-naprama-više te podržava veze više-naprama-više. Prikladniji je za kompleksnije strukture jer dopušta međusobno povezivanje entiteta, odnosno ne zahtijeva striktne odnose „roditelj“ – „dijete“, već svaki entitet može imati više „roditelja“, a kao što je bilo i u hijerarhijskom modelu „roditelj“ može imati više „djece“. Stvorio ga je Charles Bachman 1969. godine (Ullman, 1982). Uzmimo za primjer trgovinu, koja ima određene proizvode, a prodaju ih zaposlenici kupcima kroz transakcije. Neizbježno je kompliciranje modela ukoliko bismo koristili hijerarhiju, no mrežni model zbog slobodnijeg povezivanja omogućava prirodnije veze između entiteta. Tako se entiteti zaposlenici i kupci mogu povezati i s trgovinom i transakcijama te stvoriti malu mrežu međusobno povezanih entiteta. Grafikon 4 prikazuje mrežni model.



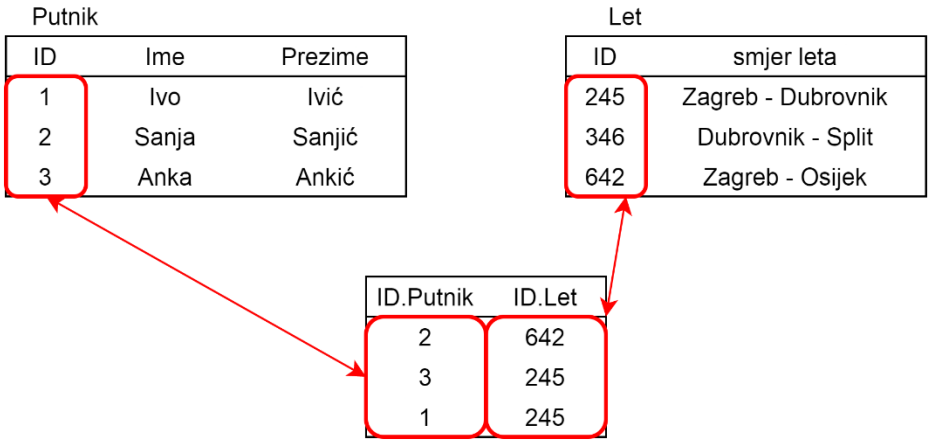
Grafikon 4- primjer mrežnog modela

2.1.3. Relacijski model

Relacijski model (engl. *relational model*) je model baza podataka koji slijedi set pravila Edgara Francka Codda, engleskog računalnog znanstvenika koji ga je izmislio i opisao krajem 1960.-tih godina (Riordan, 1999). Za razliku od prethodnih modela, u relacijskom modelu korisnik baze podataka ne bi se zamarao strukturom podataka, već bi se bavio samo onim podacima koji su mu potrebni. U relacijskom modelu podaci su organizirani u tablice s recima i stupcima koje se nazivaju relacije (engl. *relation*) povezane svima trima vrstama veza (jedan-naprama-jedan, jedan-naprama-više, više-naprama-više) te se svi upiti koje korisnik izvršava obavljaju tako što se zapravo stvara nova tablica s potrebnim podacima iz onih već postojećih (Garcia-Molina, et al., 2002).

Relacijski model uvodi novu terminologiju u svijet baza podataka. U stvaranju relacijskog modela potrebno je, naravno, identificirati entitete (tablice baze podataka) kojima opisujemo stvarnost i potom njihove atribute (stupci u tablici) koji mogu biti ključni (engl. *key*) ili ne-ključni (engl. *non-key*). Razlikovanje među atributima potrebno je zbog međusobnog povezivanja entiteta i održavanja baze podataka stabilnim okruženjem u kojem ne dolazi do slučajnih promjena podataka. Osim toga, upravo veze između atributa čine relacijski model onim što jest (Riordan, 1999). Ključni atributi su oni prema kojima se entitet razlikuje od drugih entiteta i uvijek su jedinstveni tom entitetu. Entitet može imati više ključnih atributa, odnosno ključeva (engl. *key*), a u tom slučaju jedan od njih mora biti primarni ključ (engl. *primary key*) (Varga, 2020).

Centralni pojam relacijskog modela je relacija, koja je u teoriji baza podataka isto što i relacija u matematici, tj. način kojim su dvije stvari međusobno povezane (Znanje, 2021). Preciznije, relacija opisuje vezu između atributa entiteta u bazi podataka. Samo postojanje veza između entiteta i mogućnost pohrane i povezivanja podataka koji su na različitim mjestima posebnost je relacijskog modela i razlog popularnosti tog modela još i danas. U grafikonu 5 prikazan je primjer relacijskog modela baze podataka.



Grafikon 5- primjer relacijskog modela

Model entitet-veze (engl. *entity-relationship model*, skraćeno ER model) opisao je Chen u svom članku 1976. godine. To je grafički prikaz podataka koji su međusobno povezani u sustavu kojeg promatramo, a popularna je metoda za modeliranje podataka jer je bliska korisniku te olakšava komunikaciju između projektanta baze podataka i korisnika (Chen, 1976) (Pavlić, 2011). Postoji niz postupaka kojim se iz stvarnih okolnosti dolazi do podataka koji postaju entitetima u bazi podataka, odnosno postupci apstrakcije (engl. *abstraction*) kojima se odmjerava potreba za cjelinom i detaljima te razlikuje ono što je ključno od onog što je sporedno. Prvi od takvih postupaka je klasifikacija (engl. *classification*) gdje se slični objekti, objekti koji imaju iste atribute grupiraju i predstavljaju klasom objekata. Primjerice grupiranje ljiljana i maslačaka u klasu cvijeće. Agregacija (engl. *aggregation*) je postupak apstrakcije kojim se stvara novi entitet višeg stupnja na temelju povezanosti postojećih entiteta, pr. klase zaposlenici, kupci i proizvodi bit će povezani jer su dio klase trgovina. Još jedan postupak stvaranja modela je generalizacija (engl. *generalisation*) koja je slična klasifikaciji, ali se entiteti svrstavaju pod novu klasu koja je višeg stupnja i općenitija, pr. automobil, kamion i autobus jesu entiteti općenitog entiteta vozilo (Pavlić, 2011) (Varga, 2020).

Kod relacijskog modela javlja se i pitanje redundantnosti (engl. *redundancy*) koje označava ponavljanje informacije. Na logičkoj razini nije poželjno, dok je zbog boljih performansi sustava poželjno je na fizičkoj razini sustava. No, generalno gledajući ponovljeni i loše povezani podaci samo zauzimaju fizički prostor, što se s vremenom može nakupljati i bespotrebno usporavati rad baze podataka. Iz tog se razloga u relacijskim bazama podataka provodi normalizacija (engl. *normalization*). Postojeća shema baze podataka se modificira kako bi se smanjilo ponavljanje i međuovisnost podataka. Pravila normalizacije, odnosno 6 formi osmislio je također E. F. Codd, a odnose se uglavnom na organizaciju atributa u relacijama kako bi se svi podaci jasno pohranjivali u bazi podataka (Varga, 2020).

Postoje i modeli baza podataka koji se nazivaju NoSQL (kratica za *Not only SQL*) bazama podataka. NoSQL je pristup bazama podataka koji koristi uobičajenu shemu podataka relacijskih baza podataka, ali dohvat podataka ne izvršava uz pomoć naredbi SQL jezika, o kojem će biti više riječi u idućem poglavlju. Upotreba NoSQL baza podataka stekla je svoju popularnost u eri oblaka (engl. *cloud*) i mobilnih aplikacija zbog praktičnosti svojih performansi u odnosu na tradicionalne relacijske baze podataka (Han, et al., 2011.).

2.2. Jezici za upravljanje bazama podataka

Za korištenje baza podataka na računalima potrebno je, kao i za svaki računalni proces, imati programski jezik kojim se stvaraju naredbe. Jezik koji se obično koristi u relacijskim bazama podataka je SQL (engl. *Structured Query Language*). Razvijen je u IBM-u 1970.-tih godina nakon što je Codd opisao relacijski model podataka, a budući da je bio najuspješniji pokušaj da se relacijski model objasni sintaktički (Date, 2006), još je i danas službeni standard prema odlukama Američkog nacionalnog instituta za standarde (engl. *American National Standard Institute*, skraćeno ANSI) iz 1986. godine, i Internacionalne organizacije za standardizaciju (engl. *International Organization for Standardization*, skraćeno ISO) iz 1987. godine.

Drugi jezici u počecima razvoja baza podataka, kao što je primjerice ISBL (engl. *Information System Base Language*), približavali su se maksimalnoj pretvorbi relacijske algebre koja stoji iza relacijskog modela u funkcionalnu sintaksu računalnog jezika. Jedan od takvih jezika, SQUARE, kasnije je prerastao u jezik SEQUEL (poznat i kao SQL). Razlika između ISBL-a i SQUARE-a bila je što se SQUARE proširio značajkama koje nisu svojstvene relacijskoj algebri, dok je kasnije SEQUEL riješio probleme sintakse (Ullman, 1982).

Sam SQL sastoji se od podjezika koji se na različite načine bave bazama podataka. Prvi podjezik je DDL (engl. *Data Definition Language*) kojim se stvaraju, brišu i mijenjaju objekti

i strukture u bazama podataka. Najčešće naredbe su CREATE (stvaranje nove tablicu u bazi podataka), ALTER (mijenjanje postojeće tablice u bazi podataka) i DROP (brisanje objekta iz baze podataka). Drugi podjezik je DML (engl. *Data Manipulation Language*) koji dopušta stvaranje, brisanje, promjenu i dohvat podataka, a zbog svoje sličnosti s jednostavnim engleskim jezikom poboljšava korisničko shvaćanje i korištenje sustava. Naredbe su SELECT (dohvat podataka), UPDATE (promjena podataka), INSERT (unos novih podataka) i DELETE (brisanje podataka). Slijedeći podjezik je DCL (engl. *Data Control Language*) kojim se uređuju dozvole i pristup podacima u bazi podataka te mogućnost stvaranja i manipuliranja podacima. Naredbe su GRANT (omogućuje se korisniku privilegij pristupa) i REVOKE (privilegij pristupa se oduzima). Posljednji podjezik je TCL (engl. *Transaction Control Language*) koji se koristi za kontroliranje promjena u bazi podataka napravljenih DML-om. Naredbom COMMIT sprema se dotad napravljen posao, naredbom SAVEPOINT postavlja se točka stanja baze podataka u koju je moguće vratiti se, a naredbom ROLLBACK moguće je vratiti stanje baze podataka u ono od zadnje naredbe COMMIT (Varga, 2020).

Jezicima za upravljanje bazama podataka obavljaju se između ostalog promjene, odnosno transakcije nad bazama podataka (engl. *transaction*). Transakcija je radnja ili slijed radnji koje mijenjaju ili manipuliraju podacima u bazi podataka. One su logičan niz radnji, naprimjer mali program ili slijed naredbi koje uključuju razne procese u bazama podataka (Varga, 2020).

2.3. Sustavi za upravljanje bazama podataka

Sustav za upravljanje bazama podataka, skraćeno SUBP (engl. *Database Management System*, skraćeno DBMS) je općenito vrsta softvera koji kontrolira pristup jednoj ili više baza podataka (Date, 2006). Sustav za upravljanje bazama podataka omogućuje stvaranje baze podataka, manipulaciju i pretragu nad podacima bez da se korisnika opterećuje detaljima o načinu pohrane podataka. Svaki se sustav izrađuje prema specifičnim konceptima prema kojima se izrađuje baza podataka, što je danas zamršen posao zahvaljujući kompliciranosti stvaranja relacijskog modela baza podataka.

Sukladno tome, postoje sustavi za upravljanje hijerarhijskim bazama podataka (engl. *hierarchical DBMS*), kao što je IBM-ov IMS (skraćeno od *Information Management System*) koji je i danas moguće koristiti (IBM, 2021). Mrežni model rezultirao je stvaranjem sustava za upravljanje mrežnim bazama podataka (engl. *network DBMS*), gdje opet tvrtka IBM stvara IDMS (skraćeno od *Information Data Management System*) (IBM, 2021). Iz relacijskog modela proizašli su sustavi za upravljanje relacijskim bazama podataka (engl. *Relational*

Database System, skraćeno RDBMS). Među najpopularnijim sustavima su komercijalni sustav Oracle, MySQL (*open-source* sustav koji radi na više platformi, a potpomognut je od strane Oracle-a) i MariaDB (također *open-source*).

Neke prednosti SUBP su da se lako održavaju, nude metode za pohranu i dohvat podataka, pružaju sigurnost i integritet podataka uz minimalnu redundanciju, omogućuju integraciju u programske jezike kako bi se omogućilo povezivanje baze podataka s aplikacijom ili web mjestom te automatski stvaraju sigurnosne kopije i sustave za oporavak. Općeniti nedostatci SUBP počinju od činjenice da su to obično složeni sustavi kojima je često potrebno plaćati licencu. Mnoge tvrtke svoje podatke pohranjuju u jednu bazu podataka što bi u slučaju oštećenja sustava dovelo do gubitka podataka, a sam sustav zahtijeva puno vremena za postavljanje (Date, 2004).

2.4. Centralizirane i distribuirane baze podataka

Centralizirane baze podataka (engl. *centralized database*) su postavljene na jednom sustavu i njima upravlja jedan sustav za upravljanje bazama podataka, a mogu joj pristupiti samo korisnici koji su na istoj mreži. Centraliziranu bazu podataka čini više uređaja za pohranu podataka koji su povezani na zajednički poslužitelj i to najčešće na istoj fizičkoj lokaciji. U centraliziranim bazama podataka osigurava se integracija podataka i jednostavnost upravljanja, no bilo koja greška prekida pristup bazi i tako otežava poslovanje (Date, 2004).

Distribuirane baze podataka (engl. *distributed database*) čine, isto kao i centralizirane, jedinstvenu bazu podataka, no sami podaci smješteni su na više različitih fizičkih lokacija. Kod takvih se baza pojavljuje pitanje kako očuvati integritet i konzistentnost podataka, odnosno kako sinkronizirati podatke. No, distribuirane baze podataka imaju i svoje prednosti. Kvar ili greška koji bi kod centraliziranih baza podataka doveli do nedostupnosti podataka u distribuiranim bazama podataka ne dovode do neoperabilnosti sustava jer je upravo mogućnost dohvata s više mjesta funkcija raspodjele podataka. Također, distribuirane baze podataka omogućuju veću brzinu dohvata od centraliziranih baza podataka (Date, 2004).

2.5. Standardne i mobilne baze podataka

Sve što je dosad opisano u ovom radu odnosilo se ponajviše na standardne ili tradicionalne centralizirane baze podataka koje se obrađuju i pohranjuju u nepokretnim jedinicama, odnosno stacionarnim poslužiteljima (engl. *server*). Od početka razvoja baza podataka do danas došlo je i do mnogih novih tehnoloških rješenja i otkrića koja su ušla u svakodnevnu upotrebu te se promijenio način života. Današnji život je u pokretu i tome se moralo

prilagoditi čitavo područje baza podataka, jer više nije zadovoljavajuće za informacijske potrebe svakodnevnog čovjeka da do podataka mora fizički dolaziti zbog nepokretnosti sustava. Baze podataka morale su se stoga pokrenuti i postati mobilne (Kumar, 2006), a zahvaljujući eksponencijalnom poboljšanju performansi svih mobilnih uređaja taj je proces znatno olakšan (Stapić & Vrček, 2010). Mobilna baza podataka (engl. *mobile database system*) tako je prijenosna baza podataka koja je fizički odvojena od glavnog poslužitelja baze podataka, ali ima mogućnost komuniciranja s udaljenih mjesta te preko web-a omogućuje dijeljenje podataka. Mnogo je uređaja kojima preko weba pristupamo mobilnim bazama podataka kao što su prijenosna računala, pametni mobiteli i razni drugi PDA (skraćeno od engl. *Personal Digital Assistant*) uređaji.

Za mobilne baze podataka potrebno je imati okruženje koje ih podržava. To uključuje postojanje poslužitelja baze podataka i sustav za upravljanje bazama podataka koji pruža mogućnost manipulacije podacima, udaljenu (mobilnu) bazu podataka koja upravlja podacima, mobilnu platformu baze podataka (neki prijenosni uređaj za pristup Internetu) i dvosmjernu komunikaciju između glavnog poslužitelja i mobilnog sustava za upravljanje bazama podataka. No, samo okruženje nije idealno zbog toga što mobilni uređaji nemaju jednaku procesnu snagu, istu količinu memorije niti toliku mrežnu propusnost (engl. *bandwidth*) kojeg mogu koristiti. S obzirom na to, mobilne baze podataka moraju smanjiti količinu resursa koje koriste u svom funkcioniranju (Domingos, et al., 2014). Uzimajući u obzir različite potrebe korisnika, postoji mogućnost spajanja na glavni poslužitelj baze podataka uz pomoć mobilne aplikacije i tako se omogućuje rad s podacima. Mobilne baze podataka omogućuju korisniku da preuzima podatke na svoj uređaj i da ih prenosi natrag na glavni poslužitelj, odnosno da se baza podataka iz glavnog poslužitelja replicira u mobilnoj bazi podataka (Drosatos, et al., 2006). Zbog toga moraju imati dodatne funkcionalnosti koje inače poslužitelji nemaju potrebe imati poput komunikacije s glavnim poslužiteljem, prijenos i koordinacija podataka s glavnim poslužiteljem te stvaranje prilagođenih mobilnih aplikacija za upravljanje mobilnim bazama podataka.

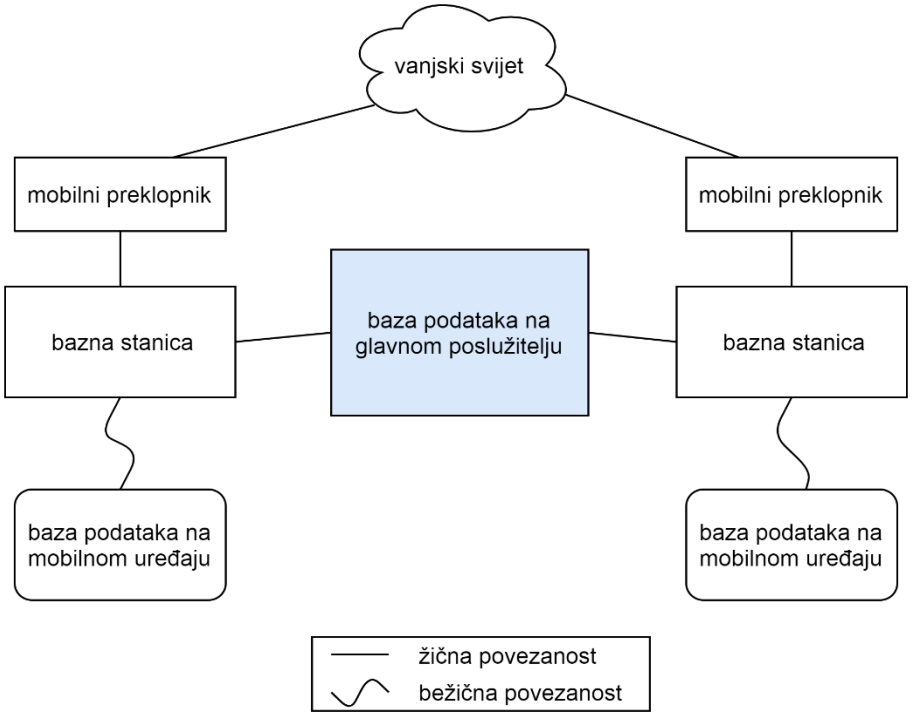
Baze podataka mogu imati arhitekturu koja zahtijeva stalnu internetsku povezanost (engl. *online*) na glavni poslužitelj čime mogu u realnom vremenu dohvaćati i mijenjati podatke. Ta je često nemoguća zbog nepouzdanog ili nepostojećeg bežičnog pristupa mreži, nepraktična zbog nedovoljne mrežne propusnosti za zahtjevnije operacije i na kraju nepoželjna zbog zaštite podataka i neisplativosti stalne povezanosti na Internet. Druga je opcija da baze podataka rade nepovezane (engl. *offline*) s ugrađenom malom mobilnom bazom podataka u

mobilni uređaj koja se periodično sinkronizira s glavnim poslužiteljem baze podataka i osigurava mogućnost manipuliranja podacima (Stapić & Vrček, 2010).

Tako su osnovna svojstva postojanja mobilne baze podataka:

- zemljopisna mobilnost – kretanje ne utječe na sposobnost baze podataka za obradu i manipulaciju podataka.
- spajanje i prekid veze – veza između glavnog poslužitelja i mobilne baze podataka nije stalna.
- sposobnost obrade podataka – zbog već navedenih razlika između stacionarnih i mobilnih uređaja, mobilni uređaji imaju manju sposobnost obrade podataka od glavnih poslužitelja baze podataka.
- bežična komunikacija – uređaj kojim se pristupa bazi podataka može komunicirati s poslužiteljem i bilo kojim drugim uređajem koji koristi istu bazu podataka u sustavu.
- skalabilnost – moguće je dodati nove mobilne uređaje u mrežu baze podataka ili postojeće izbrisati iz mreže (Kumar, 2006).

Na grafikonu 6 je prikazan pojednostavljeni primjer arhitekture mobilnih baza podataka. U samom centru nalazi se glavni poslužitelj baze podataka koji za svaku mobilnu bazu ima baznu stanicu (engl. *base station*), mobilni preklopnik (engl. *mobile switching center*) koji povezuje bazu podataka s vanjskim sustavima te dvije vrste povezanosti među sustavima: žičnu i bežičnu (Kumar, 2006).



Grafikon 6- arhitektura sustava mobilne baze podataka (Kumar, 2006)

3. Mehanizmi sinkronizacije podataka

Sinkronizacija (engl. *synchronization*) je prema svojoj rječničkoj definiciji vremensko usklađivanje dvaju ili više strojeva, uređaja, sustava ili procesa (Znanje, 2021). U pogledu baza podataka, sinkronizacija se odnosi na očuvanje integriteta, dosljednosti i ujednačenosti podataka u distribuiranim bazama podataka ili među glavnim poslužiteljem i mobilnim bazama podataka, odnosno osigurava da su isti podaci dostupni na svim jedinicama i uređajima koji koriste neku bazu podataka (Kumar, 2006).

Potrebno je razlikovati pojam sinkronizacije od pojma replikacije. Replikacija (engl. *replication*) se odnosi na kopiranje i distribuciju cjelokupne baze podataka, odnosno svih objekta i podataka, a sinkronizacija je ažuriranje naknadno promijenjenih podataka između dvije instance baze podataka, kao što su glavni poslužitelj i mobilni poslužitelj baze podataka. Replikacija baze podataka koristi se kad je potrebno osigurati bolju dostupnost podataka tako da se podijeli opterećenje na jedan dio baze podataka i nije praktičan način sinkronizacije, osobito kad se koriste mobilne baze podataka (Kovačević, 2018).

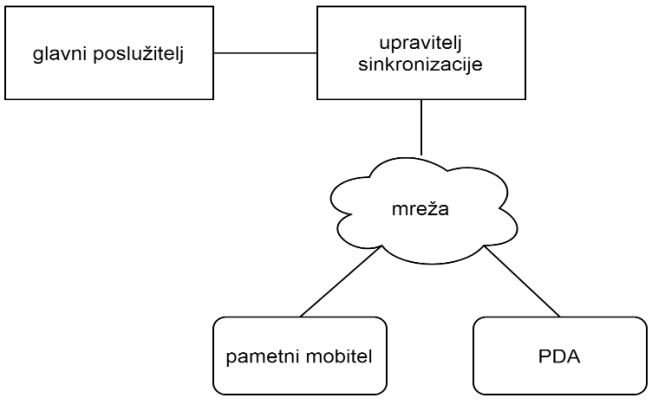
Ne postoje univerzalni sinkronizacijski mehanizmi koji se jednako koriste u svim rješenjima baza podataka, već postoje određeni parametri koje je potrebno uzeti u obzir kada se stvara sustav u kojem postoji potreba za sinkronizacijom:

- smjer sinkronizacije podataka (engl. *data synchronization direction*) – kad pogledamo bazu podataka na glavnom poslužitelju i bazu na mobilnom uređaju, postoje scenariji u kojima nije nužno sinkronizirati podatke u oba komunikacijska smjera (engl. *bi-directional synchronization*), već je važno da su podaci ažurni ili na glavnom poslužitelju baze podataka ili u mobilnoj bazi podataka.
- učestalost sinkronizacije – baze podataka se sinkroniziraju ovisno o potrebama korisnika. Sinkronizacija se može odvijati često, primjerice svaki sat ili dan, ili rjeđe, primjerice tjedno ili mjesečno. S obzirom na učestalost sinkronizacije potrebno je paziti na ugrađivanje mehanizma koji će biti odgovarajuće brzine za sustav.
- brzina sinkronizacije (engl. *database synchronization speed*) – trajanje sinkronizacije ovisi o odabiru sinkronizacijskog mehanizma. Naprimjer, značajna je razlika između sinkronizacije između dva uređaja izravno ili preko web servisa, gdje je sinkronizacija preko web servisa proces koji traje znatno dulje. Isto tako, važna je i količina podataka koja se sinkronizira.

- zahtjevi za sigurnošću podataka – mobilnim bazama podataka pristupa se preko mobilnih uređaja koji često koriste bežična rješenja za pristup Internetu, pa se sukladno sigurnosti komunikacijskog kanala može zabraniti sinkronizacija podataka ili se ugraditi dodatni sigurnosni mehanizmi tijekom prijenosa podataka.
- uključeni uređaji u sinkronizaciju – pri odabiru sinkronizacijskih mehanizama i rješenja važan je faktor ovisnost o drugim uređajima (pr. server) preko kojeg se odvija sinkronizacija, ali i vrsta uređaja koji sudjeluju u sinkronizaciji (Stapić & Vrček, 2010).

Ovisno o potrebama korisnika i parametrima moguće je odabrati između postojećih komercijalnih rješenja sustava za upravljanje bazama podataka, no često je potrebno razviti vlastitu sinkronizacijsku logiku koja zadovoljava uvjete sustava (Stapić & Vrček, 2010).

Najosnovniji mehanizmi sinkronizacije podataka su preuzimanje (engl. *download*) i učitavanje/postavljanje (engl. *upload*). U procesu sinkronizacije prvo se odvija učitavanje, odnosno podaci se iz klijentske baze podataka prenose u glavnu pa slijedi preuzimanje koje označava prijenos podataka s glavnog poslužitelja na klijentski. Tipični model sinkronizacije podataka sustava baze podataka radi s brojnim klijentima povezanima u mrežu u kojoj postoji glavni poslužitelj i mobilni uređaji koji su klijenti. Poslužitelj upravlja bazom podataka koju dijele svi klijenti putem sustava za upravljanje bazama podataka poslužitelja. Klijent se stvara tako što se preuzima dio podataka s udaljenog poslužitelja na klijentski uređaj. Kao rezultat postoji redundancija podataka, odnosno isti podaci su pohranjeni i u klijentskom uređaju i u glavnom poslužitelju baze podataka. Nakon prijenosa podataka klijent nije više nužno povezan s poslužiteljem, a sustav za upravljanje bazama podataka na klijentskom uređaju može manipulirati podacima. Sve te promjene se primjenjuju i na poslužitelj pri idućoj povezanosti poslužitelja i klijenta. Isto tako se sve promjene u poslužitelju primjenjuju i na klijenta. Glavna komponenta u procesu sinkronizacije, odnosno ono što čuva dosljednost sinkroniziranih podataka je upravitelj sinkronizacije (engl. *synchronization manager*). Upravitelj sinkronizacije dio je sinkronizacijske logike sa zadaćom kontrole nad podacima koji se mijenjaju i u kojem smjeru. Vodi evidenciju nad time koji su se podaci mijenjali od zadnje sinkronizacije i sukladno tome provodi promjene, bilo u bazi podataka glavnog poslužitelja, bilo u mobilnoj bazi podataka (Agarwal, et al., 2002) (Domingos, et al., 2014) (Singh & Hasan, 2019). Pojednostavljeni grafikon sinkronizacije prikazan je na grafikonu 7.



Grafikon 7- tipični model sinkronizacije (Domingos et al., 2014)

Sinkronizacijska logika se odnosi na postupke upravitelja sinkronizacije koji su unaprijed definirani ovisno o pravilima postupaka manipulacije s podacima u bazi podataka. U tablici 1 se nalazi primjer postupaka u različitim situacijama koje se mogu dogoditi u sinkronizaciji podataka između baze podataka na jednom klijentskom uređaju i glavnog poslužitelja (Stapić & Vrček, 2010).

Tablica 1- sinkronizacijska logika (Stapić i Vrček, 2010)

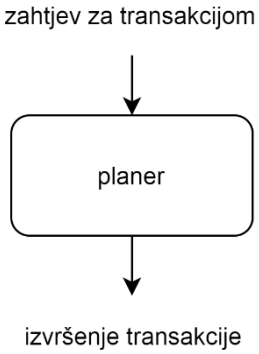
zapis klijentske baze	**zapis na poslužitelju**	**postupak u sinkronizaciji**
dodan	ne postoji	dodati zapis na poslužitelja
ne postoji	dodan	dodati zapis u klijentsku bazu podataka
obrisan	bez promjene	obrisati zapis s poslužitelja
bez promjene	obrisan	obrisati zapis s klijentske baze podataka
ne postoji	obrisan	bez postupka
obrisan	ne postoji	bez postupka
obrisan	obrisan	bez postupka
obrisan	promijenjen	vratiti i izmijeniti u klijentskoj bazi podataka
promijenjen	obrisan	vratiti i izmijeniti na poslužitelju
promijenjen	bez promjene	promijeniti zapis na poslužitelju
bez promjene	promijenjen	promijeniti zapis u klijentskoj bazi podataka
promijenjen	promijenjen	stvoriti dvije kopije koje prikazuju promjene i obavijestiti korisnika o nastalom konfliktu

Sinkronizacijska logika postaje zamršenijom ukoliko se sinkronizacija provodi između više uređaja, a taj problem „više krajnjih točaka“ rješava se primjenom koncepta brze (engl. *fast synchronization*) i spore sinkronizacije (engl. *slow synchronization*). Brza sinkronizacija provodi se kada se sinkroniziraju podaci s dva uređaja s koja su se i u zadnjoj sinkronizaciji bilježile promjene. Spora sinkronizacija odnosi se na slučajeve kad se zadnja provedena sinkronizacija nije odvila između dva uređaja u pitanju i tada je potrebno detektirati sve promjene od posljednje sinkronizacije, a u tom slučaju je i sinkronizacijska logika složenija zbog većeg broja promjena u cjelokupnom sustavu baze podataka (Agarwal, et al., 2002) (Stapić & Vrček, 2010).

3.1. Rješavanje konflikata

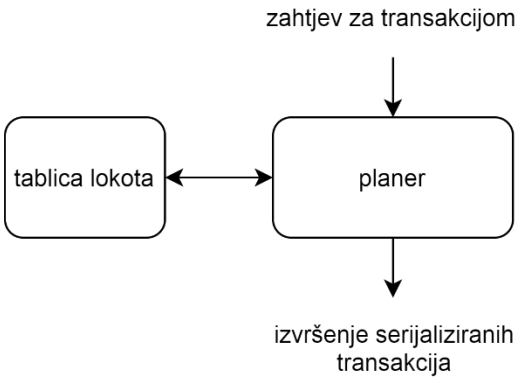
Prije sinkronizacije podataka postoje dvije (ili više) verzija podataka koji su pohranjeni u bazama podataka. Na samom početku sinkronizacije upravitelj sinkronizacije uočava promjene koje su se odvijale na uređajima i provodi sinkronizacijsku logiku, no uzmimo pretpostavku da postoji više klijentskih baza podataka u kojima se izmjenjuju podaci. U svakoj od klijentskih baza podataka može doći do različitih izmjena podataka koje se ne mogu pravilno sinkronizirati s poslužiteljem baze podataka, a tu situaciju nazivamo konfliktom ili sukobom (engl. *conflict*). Konflikti se mogu riješiti jednom od tri vrste operacija: umetanjem, brisanjem ili ažuriranjem podataka, a sve tri operacije već su nabrojane u Tablici 1 na primjeru sinkronizacijske logike (Stapić & Vrček, 2010).

U klijentskim bazama podataka koje zahtijevaju stalnu povezanost s poslužiteljem javlja se problem konkurencije (engl. *concurrency*). Promjene, odnosno transakcije u bazi podataka mogu se odvijati nad istim podacima što može ugroziti integritet baze podataka, a konkurencija se u osnovi rješava tako što se ne dopušta istovremeno odvijanje transakcija nad istim podacima s više uređaja. Ukoliko je potrebno istovremeno pristupiti podacima, sustav za upravljanje bazama podataka koristi ugrađeni planer (engl. *scheduler*) koji kontrolira redoslijed svih transakcija u bazi podataka (Garcia-Molina, et al., 2002). Planer se može grafički prikazati kao posrednik između zahtjeva za transakcijama i njihovim izvršavanjem kao što je vidljivo na grafikonu 8.



Grafikon 8- prikaz rada planera (Garcia-Molina, et al., 2002)

Najčešće će planer dopustiti odvijanje transakcije nad podacima, no postoje situacije u kojima mora odgoditi izvršavanje transakcije ili ju čak i u potpunosti otkazati, ali se na kraju sve transakcije odvijaju tako da djeluje kao da su bile obavljene jedna za drugom. Važan koncept je serijalizabilnost (engl. *serializability*) kojim se transakcije nad bazom podataka mogu razlomiti u manje naredbe koje, kad se promijene redoslijedi odvijanja manjih naredbi između više transakcija i dalje dovode do istih rezultata. Ipak, i u ovoj situaciji može doći do konflikata koji se očituju kao niz promjena u bazi podataka. U različitom redoslijedu odvijanja transakcije mogu rezultirati drugačijim promjenama u bazi podataka nego što bi to bilo da su se odvile jedna za drugom. U takvim slučajevima se u bazi podataka primjenjuje koncept konfliktne serijalizabilnosti (engl. *conflict-serializability*) koji procjenjuje može li se transakcija serijalizirati ili ne. Osnovni mehanizam planera je zaključavanje, odnosno korištenje lokota (engl. *lock*) (Garcia-Molina, et al., 2002). Lokoti se očituju kao tablica lokota na istoj razini kao i planer, kao što je vidljivo u grafikonu 9.



Grafikon 9- planer s mehanizmom lokota (Garcia-Molina, et al. 2002)

Kod zaključavanja, željene transakcije u bazama podataka zatraže otključavanje nad dijelom podataka prije vršenja promjena i nakon vršenja promjena ponovno zaključavaju taj dio podataka. Lokoti trebaju funkcionirati kao i fizički lokoti – u svakom trenutku za jedan dio podataka postoji samo jedan lokot. Uvjet koji osigurava da je redoslijed transakcija konfliktno serijalizabilan je dvofazno zaključavanje (engl. *two-phase locking*, skraćeno 2PL), odnosno da se u svakoj promjeni prvo zaključavaju neki podaci prije no što se drugi mogu otključati. Rizik i problem cijelog sustava zaključavanja koji se ne može riješiti serijalizacijom promjena u bazi podataka je slučaj kad dvije transakcije ne mogu dobiti lokot zbog druge transakcije koja se odvija i prema rasporedu čekaju zauvijek, a ta se situacija naziva stanjem mrtve točke (engl. *deadlock*) (Garcia-Molina, et al., 2002).

Drugi princip serijalizacije transakcija osim lokota kojim se u bazama podataka sprječavaju konflikti između podataka je korištenje vremenskih oznaka (engl. *timestamp*). U rješavanju konkurencije uz pomoć vremenskih oznaka, planer svakoj transakciji zadaje vremensku oznaku i planer pretpostavlja izvođenje transakcija po vremenskom redoslijedu kojim su zadane. S obzirom na to da svaki dio transakcije može drugačije trajati, u seriji transakcija može doći do čitanja prljavih podataka (engl. *dirty data*). Prljavi podaci su podaci koji se nalaze u bazi podataka, ali ne odgovaraju stvarnom stanju u trenutku transakcije. Planer stoga mora primijeniti jedno od tri pravila: dopustit će odvijanje transakcije, otkazat će transakciju i dati joj novu vremensku oznaku, ili će odgoditi odvijanje transakcije (Datta, et al., 2000) (Garcia-Molina, et al., 2002).

3.2. Mehanizmi oporavka podataka

Kao i za svaki drugi sustav, greške i kvarovi nisu isključene pojave u sustavu za upravljanje bazama podataka te je stoga potrebno osigurati podatke. Do gubitka podataka može doći zbog niza različitih situacija i ne pristupa se svim situacijama jednako: može doći do hardverskih grešaka ili nesreća (koje uključuju bilo kakva fizička oštećenja medija na kojima se pohranjuju podaci), softverskih grešaka (slučajna ili namjerna korupcija podataka) ili korisničkih grešaka (slučajno brisanje dijela podataka). Kako god da je došlo do smetnji u sustavu za upravljanje bazama podataka, svaki bi morao imati mogućnost oporavka podataka (Verhofstad, 1978).

Jedan od mehanizama oporavka podataka može biti i već navedena replikacija kojom se cjelokupna baza podataka kopira i sprema na udaljene medije za pohranu. Takve je kopije baze podataka potrebno periodično sinkronizirati kako bi u slučaju potpunog kvara glavnog poslužitelja bilo moguće vratiti podatke, no u slučaju manjih grešaka ovaj mehanizam nije

praktična opcija (Varga, 2020). Praktičnija opcija u slučaju kvarova je promatrati zapise o transakcijama u bazi podataka (engl. *log*) koji ostaju kao trag nakon obavljenih transakcija. Nakon kvara se sustav za upravljanje bazama podataka ponovno podiže i upotrebljavajući zapise primjenjuje operacije za ponavljanje (engl. *redo*) ili poništenje (engl. *undo*) nad transakcijama koje su u trenutku kvara bile u sustavu. Operacija ponavljanja dovršava obavljanje transakcije, dok operacija poništenja prekida obavljanje transakcije kako bi se očuvao integritet podataka u bazi. Postoje četiri protokola za povrat podataka ovisno o korištenju ovih operacija. *Undo-redo* koristi obje operacije za oporavak podataka, a to znači da će u slučaju da je transakcija bila aktivna u trenutku kvara ona biti poništena i ponovit će se kad bude spremna za izvršenje. *Undo-no redo* protokol oporavlja podatke tako što samo poništava aktivne transakcije u trenutku kvara. *No undo-redo* osigurava da se samo izvrše samo one transakcije koje je moguće izvršiti u trenutku oporavka, u suprotnom se odbacuju. Posljednji protokol je *no undo-no redo* koji koristi takozvane prikazne kopije (engl. *shadow copy*), odnosno podatke o provođenju promjena zajedno sa stvarnim stanjem baze podataka kako bi se oporavio sustav za upravljanje bazama podataka (Kumar, 2006).

Oporavak podataka je proces koji zahtijeva puno vremena i resursa, ali najzahtjevniji zadatak je održavanje zapisa. Kod mobilnih baza podataka je zbog smanjenih performansi mobilnih uređaja ključno imati učinkovitu i ekonomičnu shemu održavanja zapisa. U stacionarnim bazama podataka zapisi se čuvaju na glavnom poslužitelju, dok to ne mora biti slučaj kod mobilnih baza podataka, koje nisu nužno niti povezane s poslužiteljem cijelo vrijeme. Postoje tri lokacije na kojima se mogu čuvati zapisi mobilnih baza podataka, koje možemo vidjeti i na već prikazanom Grafikonu 6: mobilni uređaj na kojem se koristi baza podataka, mobilni preklopnik i bazna stanica. Čuvanje zapisa na samom uređaju zbog slabije procesne snage i memorije nije optimalno, dok se između preklopnika i bazne stanice radije bira bazna stanca zbog toga što je izravno povezana s glavnim poslužiteljem baze podataka (Kumar, 2006).

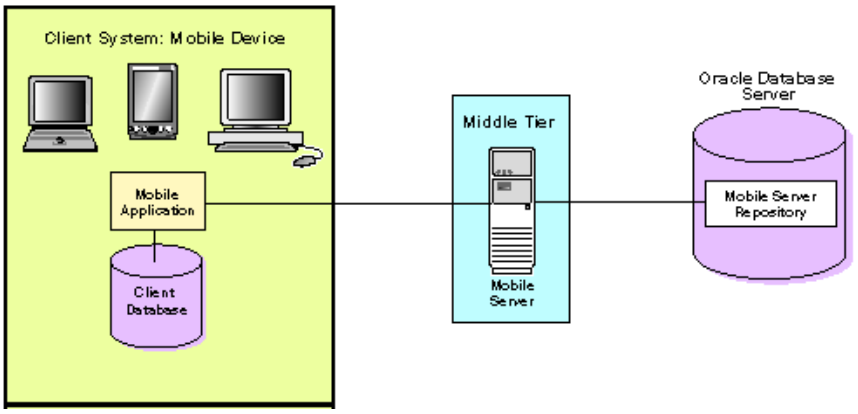
4. Profesionalna rješenja

S obzirom na niz različitih potreba korisnika u poslovanju i financijskih situacija korisnika, stvaranje održivog sustava za upravljanje podacima od samog početka nije održivo te se korisnici mogu okrenuti nizu otprije razvijenih rješenja za upravljanje bazama podataka. Korištenje komercijalnih rješenja dolazi s nizom pozitivnih točaka, kao što su brza implementacija sustava u poslovanje i manja vjerojatnost propusta u gradnji sustava. Ipak, postoji krivulja učenja, odnosno vrijeme prilagodbe korisnika postojećem sustavu od kojeg je nemoguće pobjeći, a sam sustav korisniku možda nije najfunkcionalnije rješenje, ali je tome najbliže (Stapić & Vrček, 2010). Postoje mnoga rješenja za stvaranje i održavanje relacijskih baza podataka od kojih su neki već nabrojeni u ovom radu, a u završnom dijelu rada bit će ukratko opisano par rješenja sustava baza podataka koja koriste sinkronizaciju između mobilnih uređaja i glavnog poslužitelja: dva komercijalna profesionalna rješenja prikladna za korištenje u većim organizacijama (Oracle i Microsoft) i kao kontrast jedno besplatno koje mogu koristiti manje organizacije (MobiDB). Odabrana su upravo ta rješenja zbog svoje dostupnosti i popularnosti, ali i fleksibilnosti sustava da se prilagodi potrebama klijenta.

4.1. Oracle

Oracle je tvrtka koja se bavi upravo stvaranjem softvera za upravljanje bazama podataka, sustava u oblaku i raznim poslovnim rješenjima. Stvaranje sustava za upravljanje bazama podataka u Oracle-u je započeto par godina nakon što je E. F. Codd osmislio relacijski model i otad se zove Oracle Database. Kroz godine je došlo do izuma novih tehnologija i mijenjale su se potrebe korisnika, pa su razvijena nova rješenja koja odgovaraju novim situacijama.

Jedno takvo rješenje je ono za mobilne baze podataka, Oracle Database Mobile Server (skraćeno ODMS) koji se s mobilnih uređaja povezuje na Oracle Database bazu podataka, odnosno glavni poslužitelj. Tri glavne komponente ODMS-a su repozitorij mobilnog servera na glavnom poslužitelju, mobilni klijent i posrednički mobilni poslužitelj, a međusobno su povezani kao što je prikazano na grafikonu 10.

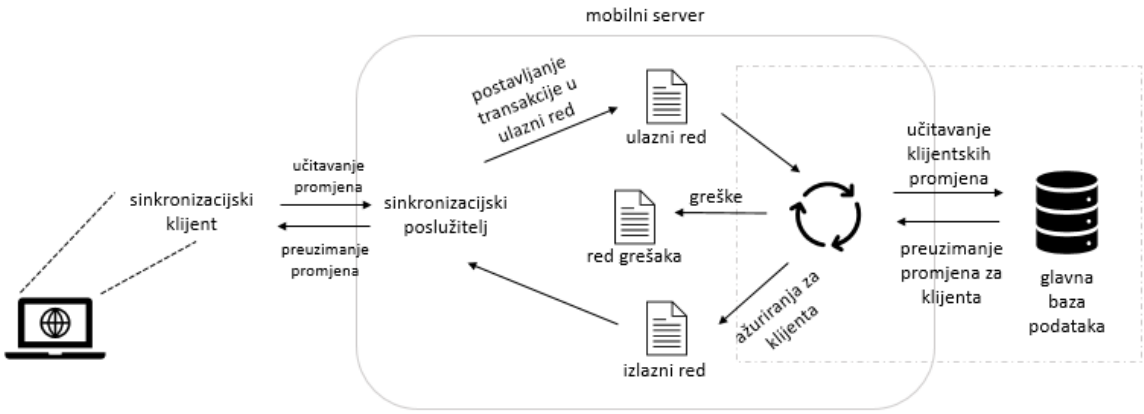


Grafikon 10- arhitektura ODMS (Oracle, 2021)

Postoje tehnički preduvjeti koje uređaj mora ispuniti kako bi bilo moguće koristiti ODMS. Za mobilni uređaj potrebna je mogućnost instalacije mobilnog klijenta, odnosno prikladni operativni sustav (Java SE, Java ME, Microsoft Windows 7 ili noviji, Android 2.2 ili noviji i mnogi drugi) i određena memorija na uređaju ovisno o platformi koja se koristi (pr. za Android to je 100KB, a za računalo s Windows operativnim sustavom 2,756KB). Za postojanje Oracle Database poslužitelja potrebno je daleko više preduvjeta: imati određeni procesor (Intel (x86), AMD64, Intel EM64T), minimalno 1GB RAM memorije, minimalno 5,35GB diskovnog prostora ne računajući dostupnu memoriju za pohranjivanje baze podataka. Potreban je prikladan operativni sustav (Windows Server sustavi te Windows XP, Vista, 7, 8 ili 8.1 operativni sustavi) te postojanje kompajlera (pr. Microsoft Visual C++ NET) i mrežnih protokola. Posrednički mobilni poslužitelj nalazi se kao aplikacija na računalu i za instalaciju aplikacije potrebno je imati hardverske preduvjete diskovnog prostora od barem 1GB, RAM memorije barem 1GB i procesor od 3GHz uz softverski preduvjet prikladnog operativnog sustava (Windows ili Linux).

Sinkronizacijski proces kojeg koristi ODMS koristi stvara red (engl. *queue*) kojim se usklađuju podaci između mobilnog klijenta i glavnog poslužitelja. Sinkronizacijski proces započinje na zahtjev korisnika ili automatski. Mobilni klijent skuplja sve promjene od zadnje sinkronizacije i pretvara ih u transakciju. Ta se transakcija učitava u posrednički mobilni poslužitelj koji stavlja transakciju u ulazni red (engl. *in queue*) iz koje se transakcija u povoljnom trenutku obavlja i sve se promjene primjenjuju na glavnom poslužitelju baze podataka. Budući da je sinkronizacija dvosmjerni proces, mobilni poslužitelj preuzima sve promjene koje je potrebno primijeniti na mobilnom klijentu i preko izlaznog reda (engl. *out queue*) izvršava potrebne promjene na bazi podataka mobilnog klijenta. S obzirom na ograničene mogućnosti mobilnih uređaja, na klijentsku bazu podataka primjenjuju se samo

one promjene koje su potrebne tom uređaju. Budući da tijekom sinkronizacije može doći do raznih grešaka, one se spremaju u red grešaka (engl. *error queue*) kojeg administrator sustava pregledava. Grafikon 11 prikazuje sinkronizacijski proces ODMS.



Grafikon 11- arhitektura sinkronizacije ODMS (Oracle, 2021)

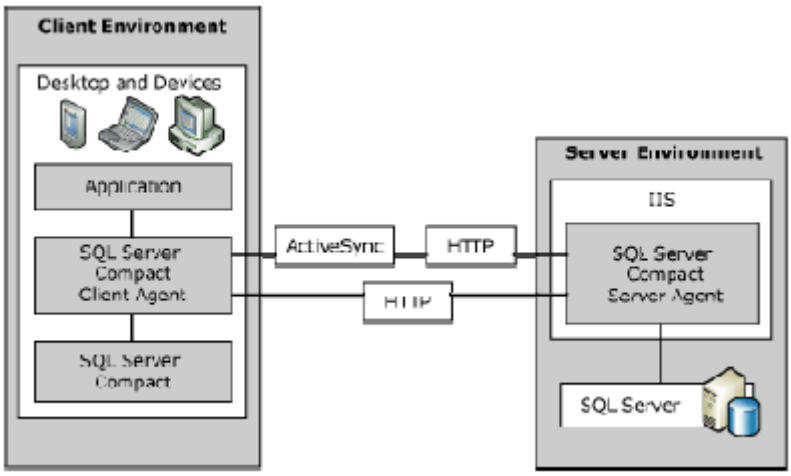
Administrator sustava u svakom trenutku može vidjeti koji se uređaji sinkroniziraju i koje se transakcije nalaze u redovima. Od tri reda koje ODMS koristi, najviše ljudske intervencije je potrebno u redu grešaka. U njega se spremaju zapisi o konfliktima i kako su oni riješeni te daju mogućnost administratoru da ih promijeni ukoliko je to potrebno. Konflikti koje sustav sam rješava nastaju kad klijent i glavni poslužitelj ažuriraju isti redak, kad stvore redak s istim primarnim ključem ili kad klijent izbriše redak kojeg glavni poslužitelj ažurira. Greške koje se zapisuju u red grešaka su i konflikti koje mobilni poslužitelj ne rješava sam, već administrator sustava odlučuje o problemu. Te se greške javljaju ukoliko glavni poslužitelj izbriše redak kojeg klijent ažurira, ukoliko klijent nije sinkroniziran s glavnim poslužiteljem ili narušava ograničenja baze podataka na glavnom poslužitelju. Administrator ima također i uvid u planer koji stvara raspored transakcija te mogućnost mijenjanja planera ili dodavanje novih transakcija u planer.

ODMS sinkronizacija dakle ne koristi lokote i tako korisnicima ne zaključava pristup bazi podataka, a korištenjem redova sinkronizacijski proces se stroga ne odvija trenutačno, već asinkronom komunikacijom između klijenta i poslužitelja kako bi se očuvao integritet podataka (Oracle Inc., 2021).

4.2. Microsoft

Microsoft je jedan od tehnoloških divova današnjice prisutan na tržištu najpoznatiji po operativnom sustavu Windows i praktičnim softverima za produktivnost Microsoft Office. Među Microsoftovim raznim softverskim rješenjima nalaze se i rješenja za baze podataka.

Iako više nije podržan, SQL Server Compact bila je mala baza podataka koju je moguće ugraditi u mobilni uređaj. Za sinkronizaciju je imala implementiran klijentski agent (engl. *client agent*) na klijentskom uređaju koji je kontrolirao vezu klijentskog uređaja prema agentu glavnog poslužitelja (engl. *server agent*) i putem HTTP protokola dohvaćao i preuzimao podatke i promjene u podacima. U grafikonu 12 vidljiva je ta sinkronizacijska shema.



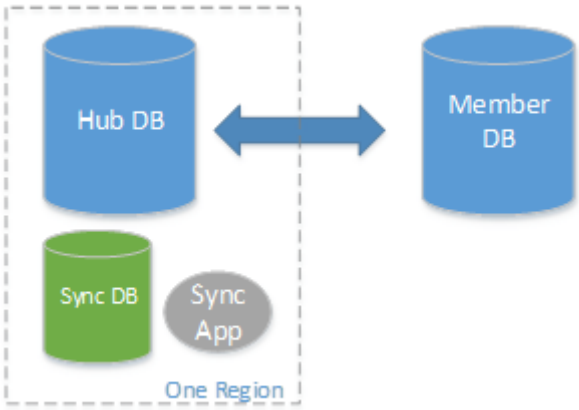
Grafikon 12- sinkronizacijska shema SQL Server Compact (Microsoft, 2011)

Tehnički preduvjeti za ovu bazu podataka bili su barem procesor Pentium 1GHz i 512MB RAM, a podržani operativni sustavi Windows XP i noviji te svi Windows Server operativni sustavi.

U procesu sinkronizacije SQL Server Compact koristi lokote i planere te tako omogućava trenutačne izmjene u bazi podataka bez konflikta među podacima. Transakcije se obavljaju redoslijedom kojim su zadane, a zapisi o transakcijama se spremaju istovremeno kako bi se u slučaju kvara očuvala konzistentnost baze podataka. SQL Server Compact rješenje je baze podataka koje ne podržava posebne uloge niti privilegije nad podacima. (Microsoft, 2011).

Što se tiče aktualnih rješenja za mobilne baze podataka i baze podataka općenito, Microsoft se okreće radu u oblacima i bazama koje rade u stalnoj mrežnoj povezanosti, odnosno online. Rješenja koja Microsoft nudi koriste tehnologiju oblaka i pohrane podataka na udaljenim serverima. Za stvaranje baza podataka razvili su Azure SQL. Koncept Azure SQL baza podataka je u tome da se podaci, odnosno baza podataka nalaze u oblaku, tj. na udaljenom

računalu. S obzirom na to, glavni poslužitelj baze podataka u ovom slučaju je instanca baze podataka koja je stvorena u Azure SQL Database okružju i naziva se središnja baza podataka (engl. *hub database*). Klijentske baze podataka nazivaju se članske baze podataka (engl. *member database*). Središnja i članske baze podataka nalaze se u sinkronizacijskoj grupi (engl. *sync group*), a sve baze podataka moraju biti u istoj regiji zbog pohrane u oblaku. Arhitektura Azure SQL prikazana je u grafikonu 13.



Grafikon 13- arhitektura sinkronizacije Azure SQL (Microsoft, 2021)

Sinkronizacija se odvija po tzv. modelu glavčine i žbice (engl. *hub and spoke model*). Središnja baza podataka zasebno se sinkronizira sa svakom članskom zasebno. Prvo se sve promjene iz središnje baze preuzimaju na člansku pa se zatim promjene s članske baze podataka učitavaju na središnju. Azure SQL nudi dvije opcije rješavanja konflikata koje se nazivaju „*hub wins"* i „*member wins"*. Kao što im imena govore, promjene ili središnje ili članskih baza podataka prepisuju promjene onih drugih, a u slučaju da je odabrana opcija „*member wins*" gleda se koja se članska baza podataka sinkronizirala prva te o tome ovisi konačan podatak nakon sinkronizacije.

Administrator središnje baze podataka ima pristup i svim transakcijskim zapisima baze podataka. Azure ovisno o odabranim opcijama stvara geografski redundantne kopije podataka i tako osigurava mogućnost povrata podataka iz bilo kojeg stanja baze podataka (Microsoft, 2021).

4.3. MobiDB Database

MobiDB baza podataka primjer je rješenja za kućnu ili upotrebu u manjim organizacijama gdje ne postoji potreba za velikim (i često skupim) rješenjima za pohranu podataka. Razlikuje se od prethodnih rješenja jer ne koristi tradicionalne sheme baze podataka i sinkronizacijske sheme. MobiDB Database je sustav upravljanja bazama podataka koji ima mogućnost

stvaranja baza podataka na relacijskom modelu, a arhitektura nije uobičajena (ne postoji glavni poslužitelj, posrednik, mobilni klijent), već se glavna baza podataka nalazi na aplikaciji na mobilnom uređaju, a posrednik kao takav niti ne postoji.

Na mobilni uređaj instalira se MobiDB aplikacija koja je glavni sustav za upravljanje bazom podataka u ovoj arhitekturi. Mogućnost sinkronizacije ostvaruje se učitavanjem baze podataka na oblak (Dropbox, Google Drive, OneDrive) i potom preuzimanjem baze podataka s oblaka na druge mobilne uređaje. Budući da se baza podataka nalazi na oblaku, promjene se primjenjuju trenutačno, ali sinkronizacija mora biti pokrenuta ručno. Stoga je na korisnicima da redovito sinkroniziraju svoje baze podataka s onom verzijom na oblaku kako bi svi radili s istim podacima.

Preduvjet za ovakvo rješenje za baze podataka je postojanje adekvatnog operativnog sustava (Windows 10, Android ili iOS sustav) i memorija za instalaciju aplikacije (Korney, 2021).

4.4. Usporedba rješenja

Sva navedena rješenja koriste relacijski model baza podataka i sustavi su koji imaju jednu glavnu bazu podataka i mobilne baze podataka koje se s njome sinkroniziraju. U ovom poglavlju sva će rješenja biti kratko uspoređena na temelju idućih karakteristika: performanse sustava, jednostavnost korištenja, sigurnost i cijena.

Što se tiče performansi sustava, najslabija je aplikacija MobiDB s vrlo jednostavnim i ograničenim mogućnostima, slijedi ukinuta Microsoftova SQL Compact Database, Azure SQL Database, a najbolje performanse ima ODMS. ODMS zahtijeva postavljanje servera koji postaju glavni poslužitelj baze podataka, a zahtjevi za klijentske uređaje ne dopuštaju da se sustav zbog mobilnih uređaja usporava. Azure SQL Database sustav je koji također ima veće tehničke zahtjeve, no ne postoji potreba za fizičkim serverima jer se baza podataka nalazi na udaljenim računalima, a korisnik plaća za njihovo korištenje.

Najjednostavnija za koristiti je aplikacija MobiDB. Sučelje je jednostavno i intuitivno, mogućnosti baze nisu komplicirane, a dokumentacija je poduprta mnogim snimkama zaslona koji olakšavaju praćenje uputa. Microsoftova rješenja imaju mnogo dokumentacije, no poprilično je nepregledna pa onemogućava lako snalaženje, ali je cijeli sustav intuitivan te je lako koristiti ga. ODMS ima ekstenzivnu dokumentaciju koja je pregledna i lako dostupna, ali sam sustav zahtijeva da korisnik prođe veliku krivulju učenja kako bi ga znao koristiti.

Daleko najsigurniji sustav je ODMS jer je cjelokupna arhitektura odvojena od vanjske mreže pa je tako teško doći do podataka u sustavu izvana. Microsoft sa svojim rješenjima u oblacima osigurava enkripciju podataka, no kao i sa svim podacima u oblaku, budući da je baza podataka na udaljenim računalima, sklonija je problemima. Isto tako, i Azure SQL Database i MobiDB za sinkronizaciju podataka između instanci baza i održavanje baze podataka zahtijevaju mrežnu povezanost koja ne ovisi o korisniku.

Na kraju, odabir korisnika svodi se na financijsku pristupačnost sustava te mogućnosti i potrebe korisnika. Kao najsigurniji sustav najboljih performansi, ODMS je od svih opisanih u ovom radu i najskuplje rješenje. Microsoftovo rješenje Azure SQL u oblaku naplaćuje se ovisno o tome koliko računalne snage koristi korisnikova baza podataka i koliko memorije baza podataka zauzima, te tako troškovi ovise o korisniku. MobiDB može biti besplatno rješenje ukoliko se koristi osnovna verzija, odnosno jedna baza podataka na jednom uređaju. Ukoliko je korisniku potreban malo kompleksniji sustav s više uređaja i mogućnošću sinkronizacije, može na godišnjoj bazi platiti sustav, a cijena ovisi o broju uređaja. (Korney, 2021) (Microsoft, 2021) (Microsoft, 2011) (Oracle Inc., 2021).

5. Primjer korištenja sinkronizacije

Sinkronizacija se ne koristi isključivo u bazama podataka korporativnih sustava kojima se svakodnevno održavaju ažurnima brojni podaci bitni za poslovanje, već i u našoj svakodnevici nevezanoj uz posao. S obzirom na to da smo okruženi mobitelima i mobilnim uređajima, ne iznenađuje pojava mnogih aplikacija kojima je cilj olakšati svakodnevicu i međusobnu komunikaciju. Između ostalih, to su aplikacije za komunikaciju i razmjenu podataka koje se nazivaju aplikacijama za čavrljanje (engl. *mobile messaging*), a stvorene su kao alternativa običnim pozivima i SMS porukama zbog visokih cijena i tarifa istih (Enderson, 2016).

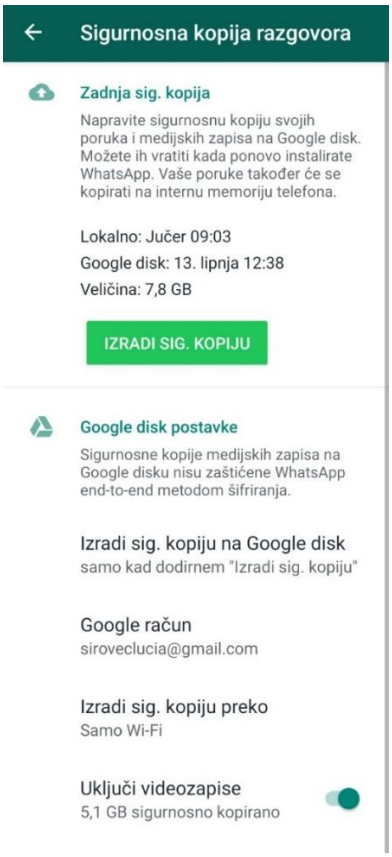
Postoji mnogo aplikacija za čavrljanje, no nisu sve jednako popularne ili raširene po svijetu, no zanimljivo je da su sve besplatne za korisnika. Primjerice, KakaoTalk je najkorištenija aplikacija za čavrljanje, no samo u Sjevernoj Koreji. Globalno gledajući najpopularnija je aplikacija WhatsApp, a slijede ju Facebook Messenger i Telegram (Similarweb LTD, 2021). Gledajući samo dvije najpopularnije, one se razlikuju po svojim značajkama koje će u nastavku rada biti ukratko predstavljene.

WhatsApp i Facebook Messenger su mobilne aplikacija za komunikaciju i jednostavno dopisivanje. Koriste internetsku vezu mobitela za razmjenu poruka i datoteka. WhatsApp ujedno osigurava potpunu sigurnost slanja poruka zahvaljujući potpunoj enkripciji. Osim razgovora između jednog pošiljatelja i primatelja, obje aplikacije podržavaju stvaranje grupa koje imaju iste funkcionalnosti kao i privatni razgovori: slanje poruka, pozivi, slanje datoteka, dijeljenje lokacije, a odnedavno postoji mogućnost brisanja poruka - prije, ali i nakon što ih je primatelj pročitao.

Aplikacija WhatsApp zahtijeva minimalne tehničke preduvjete na mobilnom uređaju (operativni sustav Android ili iOS te 180MB slobodne memorije). Ipak, prostor kojeg WhatsApp zauzima na uređaju raste s vremenom korištenja aplikacije jer se podaci razmijenjeni aplikacijom ne čuvaju na posredničkom poslužitelju. Sve poruke, fotografije, video zapisi i datoteke spremaju se na unutarnju memoriju mobilnog uređaja i tako povećavaju zahtjeve uređaja za korištenje aplikacije. Budući da aplikacija funkcionira kao baze podataka koje se nalaze na više uređaja, potrebno je te dvije kopije razmijenjenih poruka stalno sinkronizirati dodavajući nove. Po tom pitanju nema nikakvih konflikata kao što ih može biti u dosad navedenim bazama podataka: poruka pošiljatelja prolazi preko poslužitelja i dolazi na uređaj primatelja ukoliko su oba mrežno povezana i obrnuto. U slučaju

WhatsApp-a, ukoliko je pošiljatelj poruke offline, poruka se sprema i šalje kad se uređaj pošiljatelja poveže na mrežu. Nova opcija brisanja poruka trenutačno sinkronizira promjenu na uređaju primatelja tako što se poruka izbriše, ali ostaje trag da je poruka postojala.

S obzirom na to da aplikacija WhatsApp ne sprema podatke na oblaku, već na uređajima korisnika, nudi opciju sigurnosne kopije razgovora koja se zapisuje na Google Disk. Korisnik sam određuje hoće li se kopija uopće stvarati, hoće li se stvarati automatski ili ručno, a u slučaju potrebe za korištenjem kopije, korisnik će se prijaviti na svoj račun WhatsApp aplikacije i preuzeti na svoj uređaj zadnju pohranjenu kopiju. Korisnik ne može pristupiti svojim porukama ukoliko one nisu pohranjene na mobilnom uređaju ili na sigurnosnoj kopiji. Na slici 1 vidljive su opcije korisnika WhatsApp-a na Android uređaju za pohranjivanje sigurnosne kopije na Google Disk.

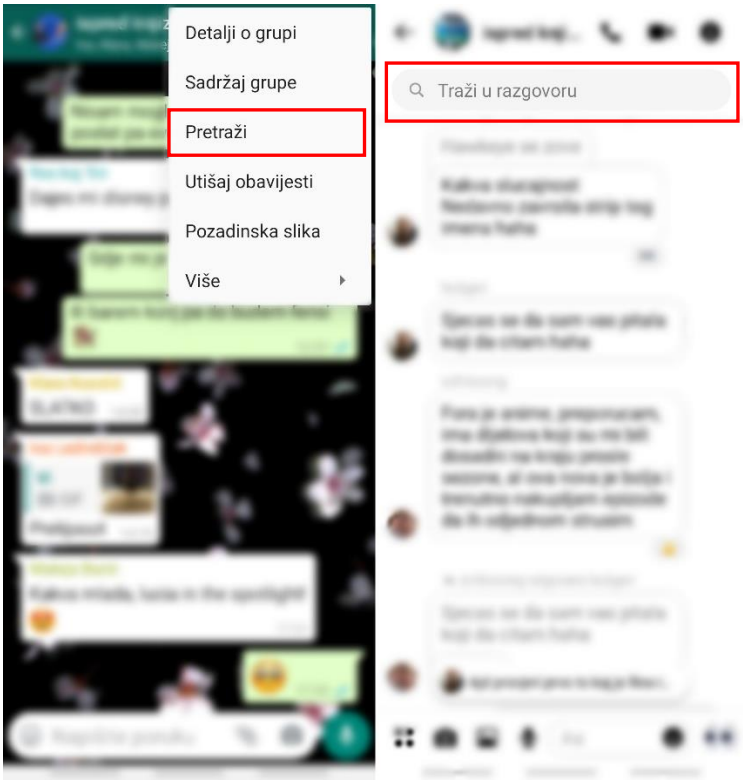


Slika 1- opcije sigurnosne kopije WhatsApp-a na Google Disku

S druge strane, aplikacija Facebook Messenger također ima minimalne tehničke zahtjeve za instalaciju (operativni sustavi Android ili iOS sa 200MB slobodne memorije ili Windows 10 s 2GB slobodne memorije), ali za razliku od WhatsApp-a sve zapise čuva na oblaku, odnosno Facebook poslužiteljima i ne zauzima dodatni prostor na uređaju nakon korištenja. Sve

promjene na oblaku se odvijaju trenutačno tijekom mrežne povezanosti korisnika, a pristup porukama moguć je sa svakog uređaja na kojem se nalazi aktivna verzija aplikacije.

Osim svega navedenog, obje aplikacije imaju mogućnost pretrage razgovora putem tražilica koje se nalaze u razgovorima, kao što je prikazano na slici 2. Tražilice pretražuju razmijenjene poruke prema nizu unesenih znakova i u slučaju WhatsAppa pretražuje po porukama koje se nalaze na uređaju, a kod Facebook Messengera pretražuje sve razmijenjene poruke između dva korisnika (Facebook, 2021) (WhatsApp LLC, 2021).



Slika 2- opcija pretraživanja poruka na aplikacijama WhatsApp i Facebook Messenger

6. Zaključak

Tijekom kratkog, ali naglog razvoja baza podataka došlo je do izmjena u potrebama manipulacijom podataka i do novih rješenja pa su se tako mijenjali modeli, sukladno tome i sustavi za njihovim upravljanjem te jezici kojima su se baze računalno stvarale. Istovremeno je došlo do eksponencijalnog razvoja tehnologije i uređaji dobivali na brzini i procesnoj snazi, a postajali manji i kompaktniji. Stoga nije neobično da se javila potreba za mobilizacijom baza podataka kako bi i one bile u pokretu. Mobilne baze podataka praktično su rješenje jer im je moguće pristupiti s gotovo svake lokacije, no performanse mobilnih uređaja znatno su manje od onih stacionarnih uređaja što je dovelo do nužnog prilagođavanja cjelokupne arhitekture baze podataka. Napredak tehnologije dopustio je i istovremeni pristup bazama podataka i njihovo mijenjanje s više lokacija. Stalna povezanost klijenata s poslužiteljem čini se kao odličan odabir dok ne dođe problema s mrežom i nedostupnošću baze podataka klijentskim uređajima, a s druge strane može se dogoditi iščitavanje krivih podataka zbog nepovezanosti s glavnim poslužiteljem. Profesionalna i komercijalna rješenja su unaprijed razvijena s vlastitim mehanizmima očuvanja i sinkronizacije podataka te su stoga vodeći su odabir tvrtki i korisnika. Sukladno potrebama, mogućnostima i zahtjevima korisnik bira jedno od rješenja, ili stavlja pred sebe izazov stvaranja novog sustava za upravljanje bazama podataka. Kao što je vidljivo iz primjera profesionalnih, ali i svakodnevnih rješenja, izgledno je da će se bez obzira na poteškoće *online* rada baze podataka spremati na oblake, tj. udaljene servere i tim putem nastavljati put mobilnosti kojeg su baze podataka prošle od stacionarnih poslužitelja do mobilnih uređaja. Samim time, ali i daljnjim razvojem tehnologije i brzine prijenosa podataka nastavit će se razvoj mehanizama očuvanja i konzistentnosti podataka.

7. Literatura

1. Agarwal, S., Starobinski, D. & Trachtenberg, A., 2002. *On the Scalability of Data Synchronization Protocols for PDAs and Mobile Devices.* s.l., IEEE.
2. Chen, P. P.-S., 1976. The Entity-Relationship Model-Towards a Unified View of Data. *ACM Transactions on Database Systems*, ožujak, pp. 9-36.
3. Date, C., 2004. *An Introduction to Database Systems.* 8. ur. s.l.:Pearson Education Inc..
4. Date, C., 2006. *The Relational Database Dictionary.* Sebastopol, CA: O'Reilly.
5. Datta, A., Son, S. H. & Kumar, V., 2000. Is a bird in the hand worth more than two in the bush? Limitations of priority cognizance in conflict resolution for firm real-time database systems. *IEEE Transactions on Computers*, svibanj, pp. 482-502.
6. Domingos, J. i dr., 2014. *Database Synchronization Model for Mobile Devices.* Barcelona, IEEE.
7. Drosatos, G. C., Efraimidis, P. S. & Karakos, A., 2006. *Secure Mobile Database Applications: A Case Study.* Xanthi, Grčka: Democritus University of Thrace.
8. Enderson, K. E., 2016. Getting acquainted with social networks and apps: WhatsApp-ening with mobile instant messaging?. *Library Hi Tech News*, kolovoz, pp. 11-15.
9. Facebook, 2021. *Messenger.* [Mrežno]
   Dostupno na: https://www.messenger.com/features
   [Pristup 28 lipanj 2021.].
10. Garcia-Molina, H., Ullman, J. D. & Widom, J., 2002. *Database Systems.* Upper Saddle River, New Jersey: Prentice-Hall, Inc..
11. Han, J., Haihong, E., Le, G. & Du, J., 2011.. *Survey on NoSQL database.* Port Elizabeth, Južna Afrika, IEEE.
12. IBM, 2021. *IBM IT Infrastructure.* [Mrežno]
    Dostupno na: https://www.ibm.com/it-infrastructure
    [Pristup 31 svibanj 2021.].

13. Kiš, M., 2002. *Englesko-hrvatski i hrvatsko-engleski INFORMATIČKI RJEČNIK.* Zagreb: Naklada Ljevak.

14. Korney, V., 2021. *MobiDB Database Documentation.* [Mrežno] Dostupno na: https://docs.mobidb.mobi/articles/index.html [Pristup 30 svibanj 2021.].

15. Kovačević, Ž., 2018. *Modeliranje, implementacija i administracija baza podataka.* Zagreb: Tehničko veleučilište u Zagrebu.

16. Kumar, V., 2006. *Mobile Database Systems.* Hoboken, New Jersey: John Wiley & Sons, Inc..

17. Microsoft, 2011. *Microsoft SQL Server Compact.* [Mrežno] Dostupno na: https://docs.microsoft.com/en-us/previous-versions/sql/compact/ [Pristup 30 svibanj 2021.].

18. Microsoft, 2021. *Microsoft Azure SQL Database.* [Mrežno] Dostupno na: https://azure.microsoft.com/en-gb/products/azure-sql/database/ [Pristup 15 lipanj 2021.].

19. Oracle Inc., 2021. *Oracle Database Mobile Server.* [Mrežno] Dostupno na: https://www.oracle.com/database/technologies/related/mobile-server.html [Pristup 15 lipanj 2021.].

20. Pavlić, M., 2011. *Oblikovanje baza podataka.* Rijeka: Odjel za informatiku, Sveučilište u Rijeci.

21. Riordan, R., 1999. *Designing Relational Database Systems.* Redmond, Washington: Microsoft Press.

22. Similarweb LTD, 2021. *similarweb Blog.* [Mrežno] Dostupno na: https://www.similarweb.com/corp/blog/research/market-research/worldwide-messaging-apps/ [Pristup 28 lipanj 2021.].

23. Singh, N. & Hasan, M. M., 2019. *Efficient Method for Data Synchronization in Mobile Database.* Allahabad, Indija, IEEE.

24. Stapić, Z. & Vrček, N., 2010. Izazovi sinkronizacije podataka između mobilnih i standardnih baza podataka. *CASE 22 - Metode i alati za razvoj poslovnih i informatičkih sustava*, pp. 107-112.

25. Technopedia Inc., 2021. *Technopedia.* [Mrežno]
Dostupno na: https://www.techopedia.com/
[Pristup 31 svibanj 2021.].

26. Ullman, J. D., 1982. *Principles of Database Systems.* Rockville, Maryand: Computer Science Press, Inc..

27. Varga, M., 2020. *Baze podataka.* 2.1 ur. Zagreb: Sveučilište u Zagrebu.

28. Verhofstad, J. S. M., 1978. Recovery Techniques for Database Systems. *ACM Computing Surveys*, Lipanj, pp. 167-195.

29. WhatsApp LLC, 2021. *WhatsApp.* [Mrežno]
Dostupno na: https://www.whatsapp.com/features/
[Pristup 28 lipanj 2021.].

30. Znanje, 2021. *Hrvatski jezični portal.* [Mrežno]
Dostupno na: https://hjp.znanje.hr/index.php?show=main
[Pristup 31 svibanj 2021.].

Popis grafikona

Popis tablica

Popis slika

Mehanizmi sinkronizacije između baza podataka

Sažetak

U radu će biti opisani osnovni koncepti baza podataka poput modela baza podataka, vrsta baza podataka, jezika i sustava za upravljanje bazama podataka. Bit će objašnjena razlika između standardnih i mobilnih baza podataka te njihova upotreba, s naglaskom na relacijske baze podataka. Analizirat će se mehanizmi sinkronizacije između baza podataka odnosno sinkronizacijska logika, mehanizmi oporavka podataka i načini rješavanja konflikata. Opisat će se funkcionalnost potpunih sinkronizacijskih mehanizama, koji će biti objašnjeni na primjerima profesionalnih poslovnih rješenja od Microsofta, Oraclea i drugih tvrtki.

Ključne riječi: baze podataka, sinkronizacija baza podataka, mobilne baze podataka

Database synchronization mechanisms

Summary

This paper will describe basic concepts of databases such as database models, database management languages and database management systems. The difference between standard and mobile databases will be explained, with an accent on relational database model. The paper will analyze database synchronization mechanisms, data recovery mechanisms and conflict resolution. Functionality of complete synchronization mechanisms systems will be broken down with examples from commercial solutions, such as Microsoft, Oracle, and others.

Keywords: database, database synchronization, mobile database