SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Siniša Kajin

TEMPORALNE BAZE PODATAKA

ZAVRŠNI RAD

Sisak, 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Siniša Kajin
Matični broj: S-42212/13-Izv
Studij: Primjena informacijske tehnologije u poslovanju

TEMPORALNE BAZE PODATAKA

ZAVRŠNI RAD

Mentor:
Doc.dr.sc. Markus Schatten

Sisak, siječanj 2017.

Sadržaj

1. Uvod

Od sredine 1960-ih godina pa sve do danas, baze podataka postale su neizbježan dio bilo kakvog informacijskog sustava. Gotovo svaki oblik podatka koji je potrebno uspoređivati ili staviti u odnos sa nekim drugim, spremljen je u bazu podataka. Poseban aspekt u svemu ima vrijeme. Pomoću zapisa koji bilježe trenutak kad je neki podatak zapisan u bazi podataka, možemo vidjeti od kada s tim podatkom možemo raspolagati i staviti ga u odnos s drugim podacima. Klasične relacijske baze podataka rade po principu da je uneseni podatak ujedno i valjan podatak. Kod temporalnih baza podataka to nije slučaj jer postoji dimenzija vremena. Podatak zapisan u bazi podataka koji je u jednom trenutku bio valjan, u drugom već ne mora biti, ili, podatak zapisan u bazi podataka koji je trenutno ne valjan, postati će valjan u zadanom (određenom) trenutku.

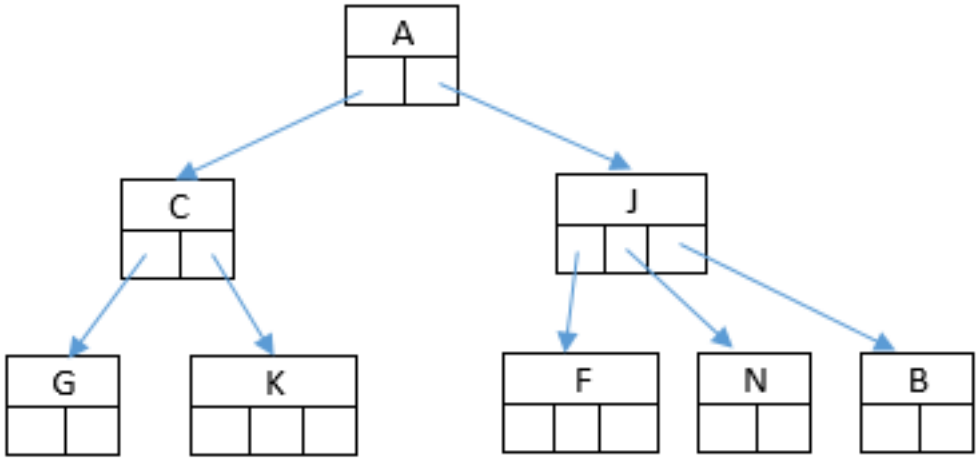
Ovakve situacije temelj su temporalnih baza podataka, a cilj ovog završnog rada bio je prikazati njihov rad, prednosti i mane kroz kreiranje temporalne baze podataka i njoj pripadajuće aplikacije, u kojoj se podaci mogu unositi, provjeravati i prikazati ovisno o proizvoljnim vremenskim parametrima za neku zadanu domenu.

2. Baze podataka

Termin „baza podataka“ opširno obuhvaća bilo kakav oblik zbirke podataka. U slučaju računalne tehnologije, ona označava digitalni zapis podataka organiziranih na prethodno definiran način te upravljan od za to predviđenog računalnog programa. Baze podataka služe nam kao temelj za kreiranje informacija pomoću kojih donosimo odluke. Prvi oblici baza podataka nastaju sredinom šezdesetih godina dvadesetog stoljeća, usporedno s razvojem direktnog pristupa zapisu podataka. Prethodni oblik zapisa i pohrane podataka bio je sekvencijalan, a obavljao se na magnetnim trakama koje su se pokazale izrazito nepraktičnima u vidu pristupa podacima u vrlo kratkom vremenskom roku. Naime, da bi pristupili određenom podatku na traci čija je duljina znala iznositi između 350 i 750 metara, ona se trebala premotavati čime se gubilo dosta vremena, a i sam postupak premotavanja nosio je određeni rizik od oštećenja zbog velikih mehaničkih naprezanja (IBM, 1962).

Dolaskom računalnih diskova i bubnjeva na scenu, baze podataka mogle su se koristiti s daleko većim potencijalom zbog toga što se vremenski rok pristupa podatku mogao dijeliti između više korisnika.

Kao navigaciju kroz podatke, prve baze podataka koristile su neki od tri načina pristupu podataka: pomoću primarnog ključa, navigacijskih veza ili čitanjem svih podataka u nizu. Naknadno je osmišljen sustav pod nazivom B-stabla, unaprijeđeni sistem crveno-crnog stablastog zapisa kod kojih čvorovi u stablu (roditelj, eng. *parent*) sadrže informacije čvorova ili listova iz niže razine a s kojima je povezan (dijete, eng. *child*). Svi ti sustavi bili su izrazito kompleksni te su tražili dugu naobrazbu korisnika kao i veliki trud pri korištenju (Bayer, 1972).



Slika 1. Primjer B-stabla (Bayer, 1972)

2.1 Relacijske baze podataka

Novi oblik baza podataka, tzv. relacijske baze podataka, nastaje 1970. godine od strane IBM-ovog zaposlenika Edgara F. Codda kao odgovor na glavni problem u starijim modelima baza podataka – pretraživanje podataka. Relacijske baze podataka temelje se na logici teorije skupova, u kojem se podaci pohranjuju pomoću relacija, prikazanih kao pravokutne tablice. Relacija predstavlja grupu entiteta: jedan red prikazuje jedan entitet, dok jedan stupac prikazuje jedan atribut entiteta. Relacija prikazuje isključivo entitete koji dijele iste atribute, bez obzira na razliku vrijednosti tih atributa. Jedan atribut mora prikazivati isti tip podatka (brojeve, tekst, logičko stanje, binarni zapis i sl.) što znači da u stupac kojem je tip podatka broj ne možemo upisivati slova, no to ne znači da brojeve ne možemo upisivati u stupce koji imaju tip podatka „tekst“. U tom slučaju se broj zapisan u tekstualno polje uzima kao tekstualni zapis (Codd, 1970).

Broj atributa jedne relacije ujedno prikazuje stupanj relacije, dok jedan entitet sa svojim atributima predstavlja n-torku. Najčešći način zapisivanja podataka podrazumijeva da ne postoje dva ista entiteta. Za relaciju u kojoj ne postoje višestruki zapisi istog entiteta i u kojem se svi atributi ne mogu dalje dijeliti kažemo da je normalizirana. Normalizacija je proces kojim primjenjujemo zadani skup pravila na sam dizajn baze podataka, točnije njezine relacije, kako bi se u što većoj mogućoj mjeri izbjeglo ponavljanje istih podataka. Bilo kakav oblik permutacija redova i stupaca ne mijenja relaciju već samo njezin prikaz (Codd, 1970).

Bitan element u relacijskoj bazi podataka predstavlja atribut (ili skup atributa) koji jednoznačno određuje n-torku, a naziva se ključem. Ako postoji više atributa sa takvim karakteristikama, jedan od njih određuje se kao primarni ključ. U slučaju da je primarni ključ skup atributa, takvi se atributi nazivaju primarnim atributima, a kako jednoznačno prikazuju n-torku, oni ne smiju biti neupisani, to jest nedefinirani (Codd, 1970).

Relacije prikazujemo na način da prvo definiramo ime relacije, a u zagradi nakon imena imenujemo atribute gdje je primarni ključ atribut podvučen punom crtom.

Ovo je primjer relacije:

KORISNIK (<u>OIB</u>, ime, prezime, godina_rođenja, …)

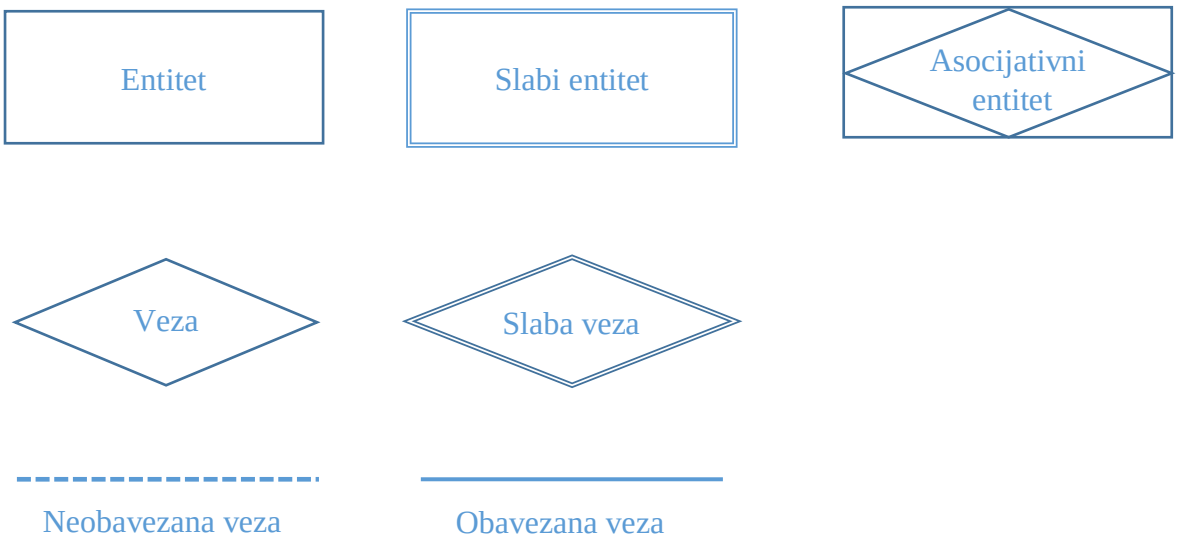
Svaki atribut entiteta ujedno postaje atribut relacije ako je takav podatak potrebno zabilježiti.

2.1.1 Tipovi entiteta i veza

Osnova svake baze podataka jest međusobna povezanost podataka. Kod relacijskih baza podataka povezanost se ostvaruje pomoću gore opisanih ključeva. Relacije koje povezujemo moraju imati zajednički atribut, stoga jednoj od relacija dodjeljujemo atribut ključ druge relacije, a takav prepisani ključ nazivamo strani ključ. Drugi način povezivanja je stvaranje nove relacije koja bi sadržavala ključeve svih relacija koje želimo međusobno povezati (Chen 1976).

Tipove entiteta dijelimo na snažne entitete, slabe entitete i asocijativne entitete. Osnovni entiteti su snažni entiteti koje možemo jednoznačno identificirati njihovim atributima. S druge strane slabi entiteti ne mogu se opisati vlastitim atributima i ovise o nekom drugom entitetu kojeg nazivamo entitet vlasnik. Slabi entitet opisujemo pomoću kombinacije identifikatora vlasničkog entiteta i dijela atributa slabog entiteta. Asocijativni entiteti su entiteti koji nastaju prilikom opisivanja veza „više prema više“ ili „jedan prema jedan“, gdje se asocijativni entiteti prikazuju kao nova relacija (Chen 1976).

Veze i uvjetnost veza između entiteta dijelimo na jake i slabe veze, te obavezne i neobavezne veze, gdje jaka veza predstavlja mogućnost postojanja veze između entiteta bez da sadrže vrijednosti njihovih primarnih ključeva, a slaba veza ovisi o entitetu-roditelju i gdje primarni ključ entiteta-djeteta sadrži elemente primarnog ključa entiteta-roditelja. Uvjetnost može biti obavezna ili neobavezna (Chen 1976).



Slika 2. Chenove oznake za relacijski model podataka (Chen, 1976)

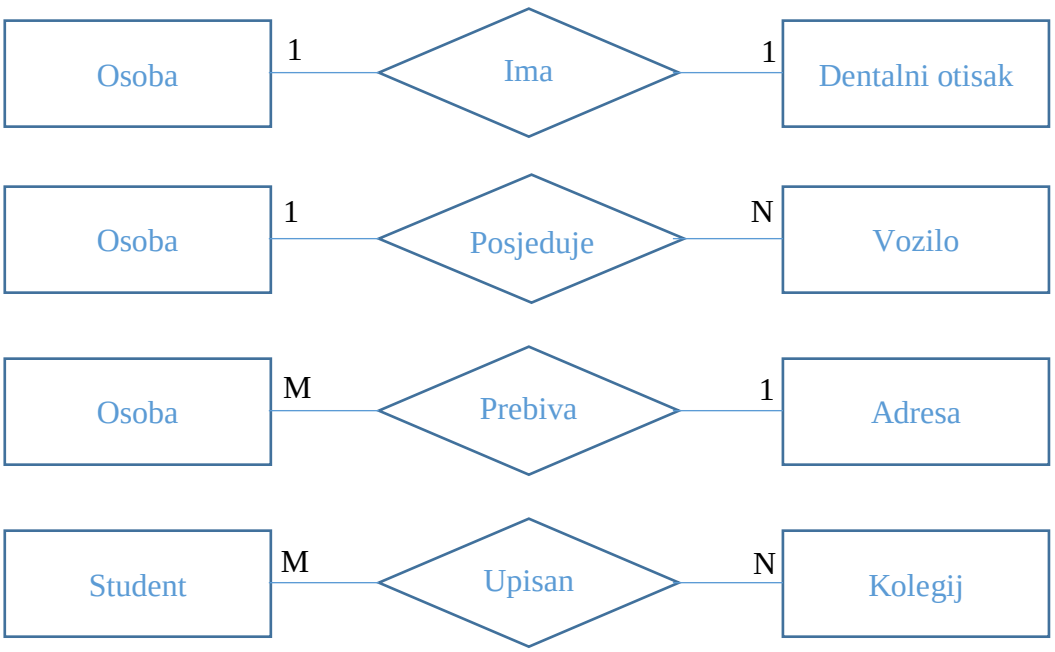
2.1.2 Kardinalnost relacija

Postoje četiri glavna tipa veza: jedan prema jedan (1:1), jedan prema više (1:N), više prema jednom (M:1) i više prema više (M:N). „Jedan prema jedan“ je tip veze u kojem su dva podatka isključivo vezani jedan uz drugoga. Takva veza prikazuje se posebnim relacijama koje opisuju tu vezu, kao na primjer vezu između osobe i njezinog dentalnog otiska.

Drugi tip veze je „jedan prema više“. Primjer tog tipa veze je osoba koja može biti vlasnik jednog ili više vozila.

Kod veze „više prema jednom“ kao primjer uzeo bih osobe i prebivališta, gdje više osoba može imati prebivalište na samo jednoj adresi.

Kao zadnji primjer veze „više prema više" mogu navesti veze studenata i kolegija. Svaki kolegij može imati više prijavljenih studenata te svaki student može imati prijavljeno više kolegija. Takve se veze ne mogu koristiti u bazama podataka te ih je potrebno prikazati vezom jedan prema više pomoću dodatne relacije (Chen 1976).



Slika 3. Primjeri kardinalnosti

3. Strukturirani jezik upita

Jedan od prvih računalnih jezika za upravljanje relacijskim bazama podataka stvoren je početkom sedamdesetih godina dvadesetog stoljeća i temeljen na relacijskom modelu C. Codda: SQL (eng. *Structured Query Language*), u prijevodu „jezik strukturiranih upita“. SQL se opisuje kao deklarativan jezik, ali sadržava neke proceduralne elemente. Sadrži jezike definiranja podataka (CREATE, ALTER, RENAME, DROP, TRUNCATE), upravljanja podacima (INSERT, UPDATE, DELETE) i kontrole podataka (GRANT, REVOKE). SQL jezik prvi put je standardiziran 1986. godine od strane ANSI (eng. *American National Standards Institute)* ustanove pod oznakom X3.135-1986 te Međunarodne organizacije za standardizaciju ISO (eng. *International Organization for Standardization)* pod oznakom ISO 9075:1987, pojednostavljeno SQL-87. Problem je nastao zbog toga što mnoštvo često korištenih i neophodnih svojstava nije bilo uključeno u standard, no, usprkos tome, taj standard postavio je temelje za korištenje SQL jezika u četiri bitna programska jezika: COBOL, FORTRAN, PL/I i Pascal. Tri godine nakon početnog izdanja, SQL se certificirao pod novim izdanjem ISO 9075:1989, tj. ANSI X3.135-1989, poznatiji pod nazivom SQL-89. Značajna razlika bila je dodavanje podrške za dva bitna programska jezika: C i Ada (Kline 2001).

X/Open konzorcij, utemeljen 1984. godine od strane nekoliko europskih proizvođača UNIX sustava, kroz ovaj period izdavanja nekoliko certificiranih verzija SQL-a izdaje svoju prvu specifikaciju SQL-a, nevezano uz ISO ili ANSI standarde, na način da je dizajn prilagodio postojećim proizvodima njihovih korisnika. No situacija nije išla na ruku X/Open konzorciju tako da se njihova specifikacija počela uvelike naslanjati na SQL-89 standard (Kline 2001).

Slijedeće izdanje SQL standarda izdalo se 1992. godine, a donijelo je novi pomak korištenjem spremljenih procedura. Ovakav pristup izvršavanja upita uvelike ubrzava rad aplikacija i općenito operacijama nad bazom podataka. Novost je uvođenje različitih razina SQL-a, Entry-SQL (koji uključuje sve temeljne elemente iz ISO 9075:1992 revizije) i Intermediate-SQL (koji uključuje dodatne mogućnosti, kao što su dinamički SQL, mogućnost udaljenog spajanja i dijagnostičke mogućnosti). Kako se sva poduzeća nisu mogla brzo i jednostavno prebaciti sa Entry-SQL na Intermediate-SQL razinu, NIST institut (*eng. National Institute of Standard and Technology*) izdaje testni okvir i novu verziju vlastitog standarda FIPS-127.2 (*eng. Federal Information Processing Standard*) i na posljetku definira Transitional-SQL razinu koja može biti korištena kao temelj za certifikaciju kroz određeni vremenski period, a kao međukorak za implementaciju Intermediate-SQL razine (Kline 2001).

Jedna od važnijih proširenih mogućnosti koje je X/Open konzorcij uveo u ovom periodu je mehanizam indeksiranja podataka. Revizija standarda iz 1992. godine donijela je slijedeće bitnije novosti:

- Definiranje novih tipova podataka: DATE, TIME, TIMESTAMP, INTERVAL, BIT string, VARCHAR string i NATIONAL CHARACTER string
- Provjera integriteta za unos podataka pomoću CHECK ograničenja
- Upotreba CASE
- Operacije UNION JOIN i NATURAL JOIN
- Dinamičko izvršavanje upita (za razliku od pripremljenog izvršavanja upita)
- Kreiranje privremenih tablica

Situacija se donekle promijenila izlaskom novog standarda 1999 godine. Sve razine SQL-a uklonile su se i postavile nove razine: osnovna razina - Core SQL:1999 i napredna razina – Enhanced SQL:1999. Razlike između Entry-SQL i Core SQL:1999 su bile male; osim što je Core SQL:1999 sadržavao sve elemente Entry-SQL razine, dodane su još neke nove mogućnosti. Proizvođači koji su razvijali baze podataka i bili u skladu sa Core SQL:1999 mogli su dodati jedan od devet dodatnih paketa mogućnosti i time bi bili u skladu sa Enhanced SQL:1999 standardom (Kline 2001).

Oznaka	Naziv	Mogućnosti
PKG001	Napredne „datetime“ opcije	- Tip podatka „interval“ - Specifikacija vremenske zone - Cijeli tip podatka „datetime“ - Neobavezan „interval“ kvalifikator
PKG002	Napredno upravljanje integritetom	- Tvrdnje za limitiranje i kontrolu (*eng. Assertions*) - Referencijalne radnje brisanja - Referencijalne radnje ažuriranja - Upravljanje ograničenjima - Pod-upiti u CHECK ograničenju - Okidači - FOR EACH STATEMENT okidač - Referencijalna radnja RESTRICT

PKG003	OLAP mogućnosti	- CUBE i ROLLUP - INTERSECT operator - Konstrukti za redove i tablice - FULL OUTER JOIN - Skalarne vrijednosti pod-upita
PKG004	Trajno pohranjeni moduli - PSM (*eng. Persistent Stored Modules*)	- Programski dodatak SQL bazi podataka koja time postaje pogodna za razvoj funkcionalno kompletnoj aplikaciji - Naredbe CASE, IF, WHILE, REPEAT, LOOP i FOR - Pohranjeni moduli - Računska potpunost - INFORMATION_SCHEMA pogledi
PKG005	Call-level sučelje – CLI (*eng. Call-level Interface*)	- Podrška za SQL Call-level sučelje: API (*eng. Application Programming Interface*) sučelje za pozivanje SQL operacija na sličan način kako funkcionira Open Database Connectivity (ODBC) standard
PKG006	Podrška za osnovne objekte	- Preučitavanje funkcija i procedura pozvanih od strane SQL-a - Korisnički tipovi podataka s jednom nasljednošću; osnovne SQL rutine na korisničkim tipovima podataka (uključujući dinamičko otpremanje) - Reference tipovi podataka - CREATE TABLE naredba - Podrška za nizove: osnovna podrška, izrazi, pretraživači, podrška za korisničke tipove nizova, podrška za nizove sa referentnim tipom - Reference za atribute i polja - Operacije za reference i uklanjanje istih

PKG007	Podrška za napredne objekte	- ALTER TABLE, ADD naredbe - Napredni korisnički tipovi podataka (uključujući opcije konstruktora, početne postavke atributa, višestruko naslijeđivanje i izraze sortiranja) - SQL funkcije i razrješenje naziva tipova - Pod tablice - ONLY naredba u upitima - Upravljanje podtipovima - Korisnički definirane CAST funkcije - UDT pretraživači - SQL rutine na korisnički zadanim tipovima kao što su funkcije identiteta i poopćeni izrazi
PKG008	Svojstva aktivne baze podataka	- Prekidači
PKG009	Podrška za SQL multimediju	- Upravljanje podacima multimedijalnog strujanja (*eng. stream*) i za kompleksne audio i video podatke

Tablica 1. Tablica paketa i njihovih opcija za proširenje Core SQL:1999 standarda (Kline 2001).

Slijedeća revizija SQL standarda izašla je 2003. godine. Promjene su vidljive i u načinu označavanja, standard se podijelio u segmente, i to na način opisan u tablici:

Oznaka	Naziv
ISO/IEC 9075-1:2003	Okvir (*eng. Framework*) (SQL/Framework)
ISO/IEC 9075-2:2003	Temelji (*eng. Foundation*) (SQL/Foundation)
ISO/IEC 9075-3:2003	Call-Level Sučelje (SQL/CLI)
ISO/IEC 9075-4:2003	Trajni pohranjeni moduli (*eng. Persistent Stored Modules*) (SQL/PSM)

ISO/IEC 9075-9:2003	Upravljanje vanjskim podacima (*eng. Management of External Data*) (SQL/MED)
ISO/IEC 9075-10:2003	Veze jezika objekata (*eng. Object Language Bindings*) (SQL/OLB)
ISO/IEC 9075-11:2003	Sheme informacija i definicija (*eng. Information and Definition Schemas*) (SQL/Schemata)
ISO/IEC 9075-13:2003	SQL rutine i tipovi pomoću Java programskog jezika (*eng. SQL Routines and Types Using the Java Programming Language*) (SQL/JRT)
ISO/IEC 9075-14:2003	XML bazirane specifikacije (*eng. XML-Related Specifications*) (SQL/XML)

Tablica 2. Segmenti SQL-2003 standarda (ISO 2003)

Značajna razlika u odnosu na prethodni standard vidljiv je i u podršci za XML specifikacije. 2006. godine izašla je nova verzija standarda koja je dodatno proširila načine korištenja SQL-a zajedno sa XML-om, prvenstveno po pitanju sistema pohrane XML podataka unutar SQL baza podataka, načina na koje se manipulira tim podacima i prikazuje podatke iz konvencionalnog dijela SQL baze podataka i XML podataka. Novost je bila i omogućavanje korištenje XQuery jezika unutar SQL naredbi kako bi se usporedno moglo pristupati XML dokumentima i SQL podacima.

SQL:2008 donosi dodatne promjene u 9075:2 segmentu standarda (eng. *Framework*), gdje su se poboljšale naredbe MERGE (prvotno uvedena u SQL:2003 standardu) i DIAGNOSTIC, te uvela TRUNCATE TABLE naredba (pomoću koje možemo obrisati sve podatke unutar tablice zadržavajući pri tom strukturu same tablice), INSTEAD OF prekidač, dodatne XQuery značajke, dijeljeni JOIN, kao i druga poboljšanja (ISO 2008).

Bitne promjene vezane uz temporalne podatke ušle su u standard 2011. godine, no taj dio obraditi ću u drugom poglavlju.

3.1 Pretraživanje podataka u bazi podataka

Podaci u bazi podataka sadržavaju sve što je uneseno do određenog trenutka. Kako bi se ti podaci mogli pretraživati po određenom kriteriju, koristimo izraz WHERE i određeni predikat (ili niz predikata) vezan uz informaciju koju pretražujemo. Bitno je napomenuti kako rezultat zadanih predikata uvijek mora biti istinit na zadani uvjet koji može biti točan, netočan ili nepoznat.

Predikate možemo prikazati u grupama: usporedbe (eng. *comparison*) ili količinske usporedbe (eng. *quantified comparison*), unutar opsega (eng. *between*), uključenosti (eng. *in*), sličnosti (eng. *like*), nedefiniranosti (eng. *null*), postojanja (eng. *exists*), jedinstvenosti (eng. *unique*), slaganja (eng. *match*) i presjeka (eng. *overlaps*) (ISO 1992).

Uz operatore definirane ISO standardom, PostgreSQL baza podataka koristi i vlastite operatore kako bi složila jednostavnije ili složenije predikate za rad sa opsezima.

Operator	Opis rada
=	jednakost
< >	nejednakost
<	manji od
>	veći od
<=	manji ili jednak
>=	veći ili jednak
@>	sadrži interval / sadrži element
<@	interval je sadržan u / element je sadržan u
&&	preklapaju se (imaju zajedničkih točaka)
<<	strogo lijevo od
>>	strogo desno od
&<	ne prelazi desnu stranu od
&>	ne prelazi lijevu stranu od
- | -	slijedi
+	unija
*	presjek
-	razlika

Tablica 3. Popis operatora za kreiranje predikata (PostgreSQL 2017)

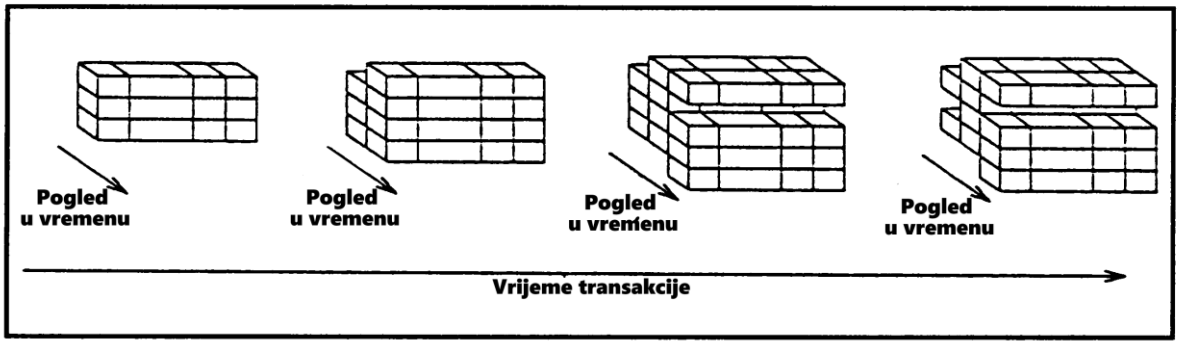
4. Temporalne baze podataka

Temporalne baze podataka razlikuju se od drugih baza podataka po jednom ključnom aspektu, a to je da se svi podaci zapisani u bazi podataka gledaju kroz dimenziju vremena, što znači da ako je neki podatak upisan u bazu podataka, ne mora nužno biti i točan (za razliku od drugih baza podataka koje zapisuju činjenične informacije koje vrijede dok zapis postoji). Temporalne baze podataka nisu vezane isključivo za relacijski tip baza podataka (npr. MySQL, PostgreSQL i dr.), već se mogu koristiti i nerelacijski tipovi kao što je NoSQL.

Ideja korištenja vremena kao atributa kod baza podataka postojala je još od 1992. godine iako tada još nije bilo podrške u standardu za takav tip podataka. Godinu dana kasnije oformljuje se odbor za bavljenje problematikom temporalnih podataka koji je 1994. godine predložio proširenje SQL standarda pod nazivom TSQL2. U konačnici se TSQL2 specifikacija sa svim komentarima izdala kao knjiga na 674 stranice 1995. godine, a definirala je kako postaviti temporalne podatke u bazama podataka (Snodgrass i dr. 1994).

Pokušaji integriranja TSQL2 u slijedeći SQL standard redom su propadali, a projekt koji se bavio integracijom TSQL2 u ISO standard se otkazao. Svejedno, slijedeći standard (SQL:1999) preuzeo je segmente TSQL2 specifikacija i ubacio ih u zaseban dio pod nazivom „SQL/Temporal“ unutar jednog od paketa.

Temporalne podatke možemo prikazivati na četiri načina: kao snimku trenutnog stanja (eng. *snapshot*), pomoću povratne relacije (eng. *rollback relation*), povijesne relacije (eng. *historical relation*) i temporalne relacije (eng. *temporal relation*). Kao najbolje rješenje, temporalne relacije koriste mehanizme povijesnih relacija i povratne relacije, na način da se prate i transakcijsko vrijeme i pogled u danom trenutku (Snodgrass, 1986).



Slika 4. Grafički prikaz temporalne relacije (Snodgrass, 1986)

4.1 ISO/IEC 9075:2011

Bitna revizija ISO standarda za SQL jezik je ISO/IEC 9075:2011 koja je definirala nove elemente vezane uz temporalne baze podataka. Po principu povijesne relacije i povratne relacije uvode se slijedeći elementi: dva atributa kao početak i kraj nekog imenovanog perioda sa principom otvoreno-zatvorene semantike, točnije mogućnošću uključivanja ili isključivanja zadanog početnog ili završnog trenutka u sam opseg perioda, te sistemski verzionirane tablice koje baza podataka sama ažurira. Uvode se novi temporalni predikati (CONTAINS, OVERLAPS, EQUALS, PRECEDES, SUCCEDS, IMMEDIATELY PRECEDES i IMMEDIATELY SUCCEDS) kao izmijenjena verzija algebre Allenovih intervala iz 1983. godine pomoću kojih možemo objasniti trinaest mogućih relacija između dva intervala (Allen 1983).

Relacija	Ilustracija	Interpretacija	SQL predikat
X < Y Y > X	X Y	X se zbiva prije Y	SUCCEDS PRECEDES
X m Y Y mi X	X Y	X se sastaje sa Y (i znači inverzno)	IMMEDIATELY SUCCEDS IMMEDIATELY PRECEDES
X o Y Y oi X	X Y	X se preklapa sa Y	OVERLAPS
X s Y Y si X	X Y	X počinje kad i Y	CONTAINS, OVERLAPS
X d Y Y di X	X Y	X traje u intervalu Y	CONTAINS, OVERLAPS
X f Y Y fi X	X Y	X završava kad i Y	CONTAINS, OVERLAPS
X = Y	X Y	X je jednak Y	CONTAINS, EQUALS, OVERLAPS

Tablica 4. Algebra Allenovih intervala po ISO/IEC 9075:2011 standardu (Allen 1983; ISO 2011)

Definiraju se nova dva tipa tablica:

- Tablice aplikativnih vremenskih intervala tj. tablice ispravnosti (eng. *valid time tables*)
- Sistemski verzionirane tablice tj. transakcijske tablice (eng. *transaction time tables*)

Kod tablice aplikativnih vremenskih intervala koristimo „PERIOD FOR“ oznaku dok se kod sistemski verzioniranih tablica koristi oznaka „PERIOD FOR SYSTEM_TIME“ uz „WITH SYSTEM VERSIONING“ modifikator. Sistemski verzionirane tablice automatski se ažuriraju, a ograničenja nije potrebno postavljati.

Dodatno se uvode temporalni primarni ključevi koji koriste aplikativne vremenske intervale gdje možemo postaviti ograničenje „WITHOUT OVERLAPS“ (bez preklapanja). Uvodi se ograničenje temporalnog referencijalnog integriteta, kao i upiti prema sistemski verzioniranim tablicama, pomoću vremenskih isječaka i sekvenci izrazima „AS OF SYSTEM TIME“ i „VERSIONS BETWEEN SYSTEM TIME … AND ...“.

Tablice u kojima se zajedno koriste sistemski verzionirano i aplikacijsko vrijeme nazivaju se bitemporalnim tablicama (ISO 2011).

4.2 Zapis intervala u temporalnim bazama podataka

Intervale u temporalnim bazama podataka zapisujemo pomoću nekoliko tipova podataka: DATE (datum), TIME (vrijeme), TIMETZ (vrijeme sa vremenskom zonom), TIMESTAMP (datum i vrijeme kao jedna varijabla) i TIMESTAMPTZ (isto kao TIMESTAMP uz dodatak vremenske zone). Intervali koriste jedan od gore navedenih tipova podataka kao zapis početnih i završnih vrijednosti. Ako neka vrijednost nije unesena, znači da traje beskonačno. Možemo razlikovati dva stanja beskonačnosti, beskonačnost u prošlost i beskonačnost u budućnost. Beskonačnost u prošlost označavamo sa „-∞“ dok beskonačnost u budućnost označavamo sa „+∞“. Takav zapis koristimo zato što je takav podatak u tom trenutku ispravan i vrijedi do slijedeće izmjene, a kako ne znamo kad i hoće li se izmjena dogoditi; pretpostavljamo da je podatak upisan trajno, što znači da mu je u tom trenutku i vrijeme trajanja

beskonačno. U slučaju prošlosti koristimo oznaku beskonačno ako nemamo informaciju o početnom trenutku nekog unosa.

Zagrade kod upisa opisuju uzima li se navedeni datum u obzir kod intervala ili ne. Za unesene podatke koji su omeđeni uglatim zagradama „[ ]“ kažemo da ulaze u interval, dok za podatke omeđene oblim zagradama „( )“ kažemo da ne ulaze u interval. Temporalne baze podataka najčešće koriste miješani oblik zapisa intervala u kojem uključuju početni datum intervala dok završni ne uključuju. Ako unosimo podatak koji uključuje zadnji datum, upravljačka aplikacija baze podataka - DBMS (eng. *data base management system*) automatski mijenja zapis tako da unosi slijedeću najbližu vrijednost i isključuje ju iz intervala.

4.3 Primjeri korištenja temporalnih baza podataka

Kao prvi primjer možemo uzeti evidenciju prebivališta osobe. U tom slučaju, relacija bi sadržavala OIB, ime i prezime, adresu prebivališta i sistemski vremenski interval. Kada se prvi put unese prebivalište osobe, unos u bazu podataka izgleda ovako:

OIB	**Ime i prezime**	**Adresa**	**Sistemski interval**
12345678901	Ivan Horvat	Ilica 10, Zagreb	[2001-10-01,)

Kako nije unesen završni datum intervala, pretpostavljamo da entitet vrijedi trajno. U određenom trenutku može doći do promjene, odnosno, osoba se preselila na novu adresu i samim time dobiva novo prebivalište. Unosimo novi zapis na način da ažuriramo postojeći unos za navedenog korisnika. U tom trenutku DBMS radi kopiju postojećeg zapisa, unosi završni datum intervala te kreira novi unos sa novim podacima i novim početnim datumom.

OIB	**Ime i prezime**	**Adresa**	**Sistemski interval**
12345678901	Ivan Horvat	Vlaška 5, Zagreb	[2002-02-15,)
12345678901	Ivan Horvat	Ilica 10, Zagreb	[2001-10-01,2002-02-15)

Kad bi korisnik napustio državu te se odjavio, podatak bi jednostavno obrisali čime bi DBMS kopirao prethodni unos te postavio završni datum intervala bez unošenja novih informacija u bazu podataka.

Drugi primjer je enciklopedija Wikipedija. Promjene podataka bilježe se po istom principu u bazi podataka, a ono što je drugačije je da ti podaci ne moraju biti jednostavni kao u prethodnom primjeru, već se koristi XML zapis kao podatak što znači da je kompletan dio nekog članka sadržan u polju određenog atributa. Kao treći primjer možemo uzeti prodaju ulaznica za neki festival. U ovom slučaju možemo koristiti aplikacijske vremenske intervale kod kojih definiramo u kojem periodu će biti moguće iskoristiti ulaznicu.

Broj ulaznice	**Naziv festivala**	**Datum prodaje**	**Trajanje karte**
1234567	Barokni festival	2015-10-20	[2016-02-15 08:00:00.000, 2016-02-17 23:00:00.000)
5678901	Barokni festival	2015-11-02	[2016-02-15 08:00:00.000, [2016-02-15 23:00:00.000)

Kao što se vidi, u prethodnoj tablici kao vrijednost intervala koristim tip podataka „TIMESTAMP“ gdje ulaznica pod brojem „1234567“ vrijedi kroz vremenski period od 15. veljače 2016. od 8 sati, do 17. veljače 2016. do 23 sata, što znači da bilo tko može takvu kartu iskoristiti u navedenom vremenskom periodu (uključujući i vremenski podatak).

5. Projekt „Sigurna hrana“

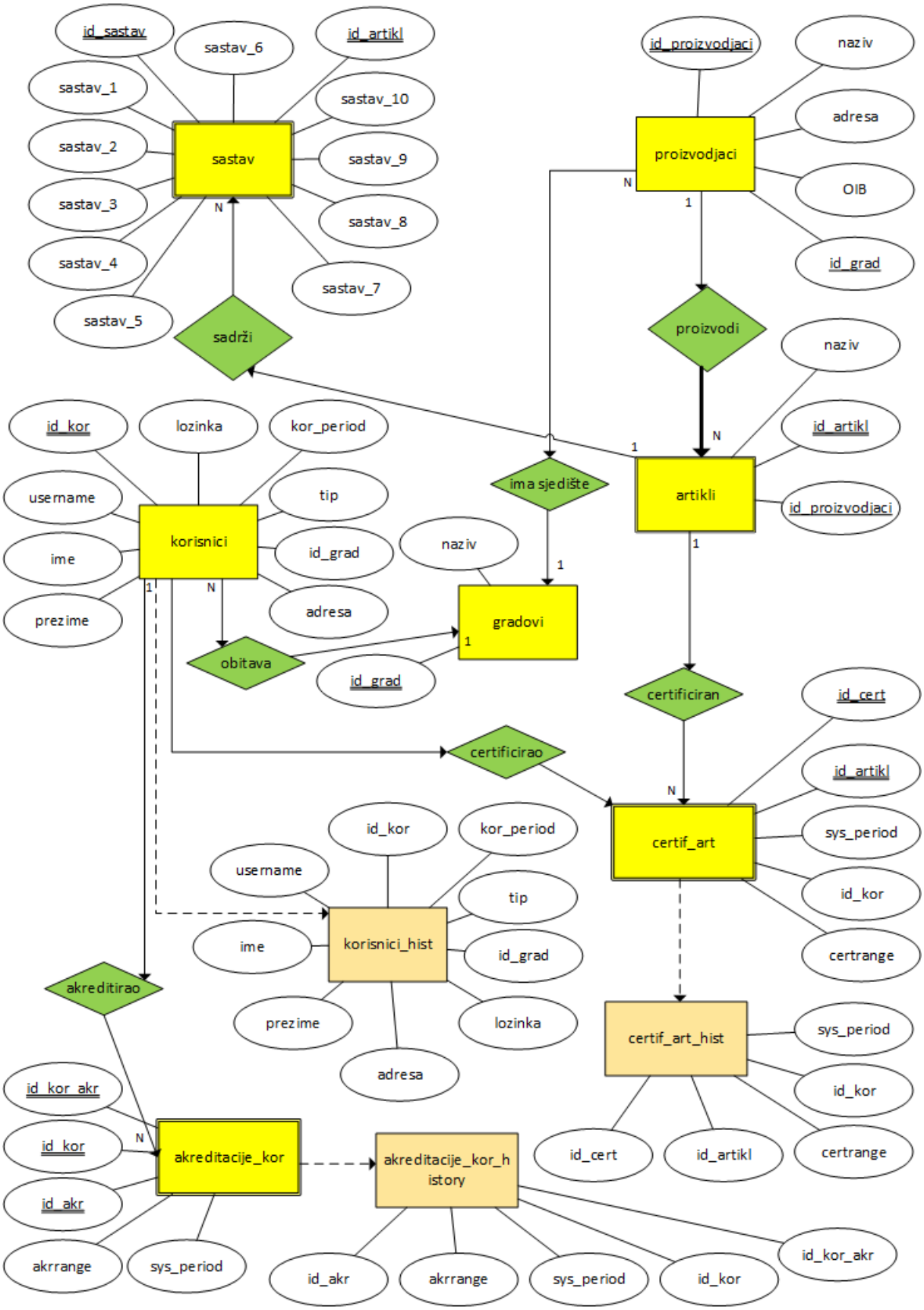
Kako bi prikazao rad temporalne baze podataka u praksi, odlučio sam napraviti web aplikaciju koja bi prikazivala razne artikle i njihove detalje, kojima bi određena osoba, koja ima mogućnost certificiranja, dodavala vremenski period u kojem je artikl certificiran. Dakle, domena koju će temporalna baza podataka opisivati uključuje certificirane artikle, detalje tih artikala i korisnike koji te artikle mogu certificirati.

Prvo sam kreirao ERA model za bazu podataka. S obzirom da sama aplikacija nije produkcijskog karaktera već konceptualna, izostavio sam veliki broj elemenata koji bi bili neophodni kako bi se ovakav tip baze podataka mogao komercijalno koristiti.

Za bazu podataka odabrao sam aplikaciju PostgreSQL koja je bazirana na otvorenom kodu i lako se nadograđuje ovisno o korisničkim potrebama. Koristio sam verziju 9.5 baze podataka (aktualne u tom trenutku), i upotrijebio proširenje „temporal tables“ jer baza podataka sama po sebi nema podršku za temporalne podatke.

Prvi problem s kojim sam se susreo bio je nekompatibilnost instalacije proširenja sa programom baze podataka. Pregledavši detalje greške, uočio sam da proširenje traži bazu podataka na specifičnoj putanji na disku gdje moja instalacija baze podataka ne pohranjuje ništa. Kako postoje dva instalacijska paketa PostgreSQL aplikacije, odlučio sam instalirati drugu i provjeriti pohranjuje li ta instalacija aplikaciju po specifikaciji proširenja. Srećom, PostgreSQL program instalirao se na putanju koja je bila preduvjet proširenja, pa sam nakon reinstalacije pokušao nanovo instalirati proširenje. I u ovom slučaju došlo je do greške s instalacijom: PostgreSQL javlja da proširenje nije kompatibilno. Provjerom na službenoj stranici proširenja „temporal tables“ uvjerio sam se da sam preuzeo zadnju verziju proširenja (verzija 1.1.0) – ali ne za moju verziju PostgreSQL aplikacije. No, na jednom od foruma pronašao sam verziju koja je kompilirana za verziju 9.5 te sam ju uspješno instalirao.

Tip podataka koji se koriste u proširenju „temporal tables“ su vrijeme i datum sa vremenskom zonom (tstzrange), a kako je taj tip podatka direktno podržan od strane aplikacije PostgreSQL, sve je bilo spremno za izradu projektne baze podataka. Proučio sam na koji način funkcionira instalirano proširenje i zaključio da je upotreba jednostavna: bitno je napraviti okidače koji će prilikom ažuriranja ili brisanja podataka pred samu radnju prepisivati trenutne podatke u zadanu povijesnu tablicu.



Slika 5. ERA model baze podataka "Sigurna hrana"

Za kreiranje tablica i atributa koristio sam konzolnu aplikaciju i aplikaciju sa grafičkim sučeljem koja dolazi u paketu sa instalacijom PostgreSQL paketa, pgAdmin 3. I u ovom slučaju došlo je do problema oko unosa podataka. Ako se koristi konzola za unos podataka, kompletan rad mora se izvršavati unoseći naredbe u kojima se može potkrasti greška ako korisnik nije pažljiv, a takvu je grešku kasnije teže pronaći i ispraviti. U slučaju korištenja aplikacije sa grafičkim sučeljem, ako se u nazivima tablica ili atributa koriste velika slova, aplikacija eksplicitno unosi naziv sa velikim slovom (ovo nije slučaj sa konzolnim unosom!) i to stvara dodatne probleme prilikom pisanja upita, bilo u konzolnoj ili razvojnoj aplikaciji.

Prvi korak nakon instalacije programskih elemenata je kreiranje korisničkog imena i lozinke, zatim kreiranje baze podataka pod imenom „sigurna_hrana“ i dodavanje „temporal_tables“ proširenja u bazu podataka.

5.1 Izrada tablica i okidača

Veliku većinu tablica radio sam u terminalnoj aplikaciji. Kako ne bi došlo do problema vezanih uz korištenje hrvatskih znakova, morao sam prije svakog pokretanja konzolne aplikacije „psql.exe“ postavljati jezične postavke na kod 1250 pomoću naredbe „chcp 1250“.

Prva tablica koju sam kreirao je „gradovi“ s obzirom da ne koristi vanjske ključeve.

```
sigurna_hrana=# CREATE TABLE gradovi ( id_grad serial NOT NULL PRIMARY
KEY, naziv text );
```

Prethodna naredba kreira tablicu „gradovi“ koja sadrži atribut „id_grad“ koji je ujedno i primarni ključ definiran kao brojčanik (tip podatka je cijeli broj, ali naredba dodatno kreira sekvencu koju naziva po sistemu „naziv tablice_naziv atributa_seq“ te joj pridružuje početnu vrijednost, maksimalnu vrijednost i vrijednost kojom će se povećavati) i atribut naziv koji ima tip podatka tekst (pohrana samih naziva gradova).

Slijedeća tablica koju sam kreirao je „proizvodjaci“.

```
sigurna_hrana=# CREATE TABLE proizvodjaci ( id_proizvodjaci serial NOT
NULL PRIMARY KEY, naziv text, adresa text, id_grad integer references
gradovi(id_grad), „OIB“ text NOT NULL);
```

Kod ove tablice zaboravio sam odmah na početku dodati atribut „OIB“ kojeg sam kasnije dodao pomoću pgAdmin 3 aplikacije. Zbog toga se taj atribut mora u svakom upitu pisati pod navodnicima zbog prije navedenog ponašanja kod kreiranja atributa iz te aplikacije. Tu se prvi put pojavljuje vanjski ključ nad atributom „id_grad“ prema tablici „gradovi“.

Iduća tablica je „artikli“.

```
sigurna_hrana=# CREATE TABLE artikli ( id_artikl serial NOT NULL
PRIMARY KEY, naziv text, id_proizvodjaci integer references
proizvodjaci(id_proizvodjaci) );
```

Bitniji detalj tablice „artikli“ je vanjski ključ nad atributom „id_proizvodjaci“ koji se povezuje na tablicu „proizvodjaci“.

Kako bi zaokružio cjelinu vezanu uz artikle kreirao sam tablicu „sastav“.

```
sigurna_hrana=# CREATE TABLE sastav ( id_sastav serial NOT NULL
PRIMARY KEY, id_artikl integer references artikli(id_artikl), sastav_1
text, sastav_2 text, sastav_3 text, sastav_4 text, sastav_5 text,
sastav_6 text, sastav_7 text, sastav_8 text, sastav_9 text, sastav_10
text );
```

Tablica „sastav“ uz vanjski ključ nad atributom „id_artikl“ prema tablici „artikli“ sadrži i deset atributa kojima se opisuje sastav nekog artikla. Na ovakav koncept odlučio sam se zbog jednostavnosti implementacije i zbog lakšeg prikaza podataka, iako se informacija o sastavu mogla zapisati u obliku XML-a i pohraniti u za to definirani atribut, što bi samo po sebi bilo daleko optimalnije.

Na red su došle tablice koje će biti uparene sa temporalnim tablicama. Slijedeća tablica koju sam kreirao je „korisnici“. Kako bih bilježio promjene nad tablicom postavio sam temporalni sustav koji će svu povijest bilježiti u za to određenoj tablici.

```
sigurna_hrana=# CREATE TABLE korisnici ( id_kor serial NOT NULL
PRIMARY KEY, id_grad integer references gradovi(id_grad), ime text,
prezime text, adresa text, tip integer, lozinka text, kor_period
tstzrange );
```

Kako je ovo prva tablica koja koristi temporalnu mehaniku, morao sam dodati atribut „kor_period“ koji ima tip podatka tstzrange (vremenski opseg koji koristi vremensku zonu). Nakon kreiranja tablice napravio sam odmah kopiju te iste tablice pod drugim imenom za pohranu promjena.

```
sigurna_hrana=# CREATE TABLE korisnici_hist (LIKE korisnici);
```

Nakon toga sam jednostavno kreirao okidač koji upisuje prethodni zapis kod bilo kakve promjene u tablici „korisnici“ u tablicu „korisnici_hist“.

```
sigurna_hrana=# CREATE TRIGGER ver_okidac
BEFORE INSERT OR UPDATE OR DELETE
ON korisnici
FOR EACH ROW
EXECUTE PROCEDURE versioning('kor_period', 'korisnici_hist', 'true');
```

Ovaj okidač jednostavno pokreće proceduru „versioning“ koja je sastavni dio proširenja „temporal_tables“. Dodatni elementi koji se moraju deklarirati su atribut kojim pratimo promjene (u našem slučaju „kor_period“) i tablicu u koju će se zapisivati sve informacije koje su se promijenile u tablici „korisnici“. U slučaju zapisa, promjene ili brisanja, prekidač pokreće proceduru.

Slijedeća tablica je „akreditacije_kor“. U ovu tablicu unose se akreditacije korisnicima koji mogu davati status certificiranosti pojedinim artiklima.

```
sigurna_hrana=# CREATE TABLE akreditacije_kor ( id_akr serial NOT
NULL, id_kor_akr integer NOT NULL, id_kor integer references
korisnici(id_kor), sys_period tstzrange, akrrange daterange, PRIMARY
KEY(id_akr, id_kor_akr, id_kor) );
```

Ova tablica za promjenu koristi kompozitni primarni ključ, točnije ključ koji se sastoji od tri atributa: „id_art“, „id_kor_akr“ i „id_kor“. Novost kod ove tablice je aplikacijski temporalni atribut „akrrange“ kojim ne barataju niti upravljačka aplikacija baze podataka niti proširenje, već se kontrolira isključivo kroz korisničku aplikaciju. Atribut „akrrange“ je tip podatka „datumski period“ (eng. *daterange*) gdje se unose početni i završni datum perioda u kojem korisnik ima pravo dodjeljivati certifikate artiklima. Atribut „sys_period“ je klasični atribut temporalnog proširenja kojim bilježimo promjene u kopiji tablice pod imenom „akreditacije_kor_hist“.

```
sigurna_hrana=# CREATE TABLE akreditacije_kor_history (LIKE
akreditacije_kor);
```

Kao i kod prethodnog seta tablica, i ovdje sam kreirao okidač koji će ažurirati tablicu sa povijesnim podacima.

```
sigurna_hrana=# CREATE TRIGGER ver_okidac
  BEFORE INSERT OR UPDATE OR DELETE
  ON akreditacije_kor
  FOR EACH ROW
  EXECUTE PROCEDURE versioning('akr_period',
'akreditacije_kor_history', 'true');
```

Zadnji set tablica u bazi podataka koriste se za bilježenje certifikata artikala i povijesti promjena tih podataka.

Kao i kod prethodnih tablica, koristio sam aplikacijski temporalni atribut i sistemski temporalni atribut.

```
sigurna_hrana=# CREATE TABLE certif_art ( id_cert serial NOT NULL,
id_artikl integer references artikli(id_artikl), sys_period tstzrange,
certrange daterange, PRIMARY KEY(id_cert, id_artikl) );
```

Atribut „certrange“ predstavlja aplikacijski temporalni atribut pomoću kojeg zapisujem period u kojem je artikl certificiran, a atribut „sys_period“ sistemski temporalni atribut koji bilježi vremena promjena u kopiji tablice za pohranu povijesnih podataka. I ova tablica zahtjeva okidač za praćenje promjena podataka.

```
sigurna_hrana=# CREATE TABLE certif_art_hist (LIKE certif_art); CREATE
sigurna_hrana=# TRIGGER ver_okidac
  BEFORE INSERT OR UPDATE OR DELETE
  ON certif_art
  FOR EACH ROW
  EXECUTE PROCEDURE versioning('sys_period', 'certif_art_hist', 'true');
```

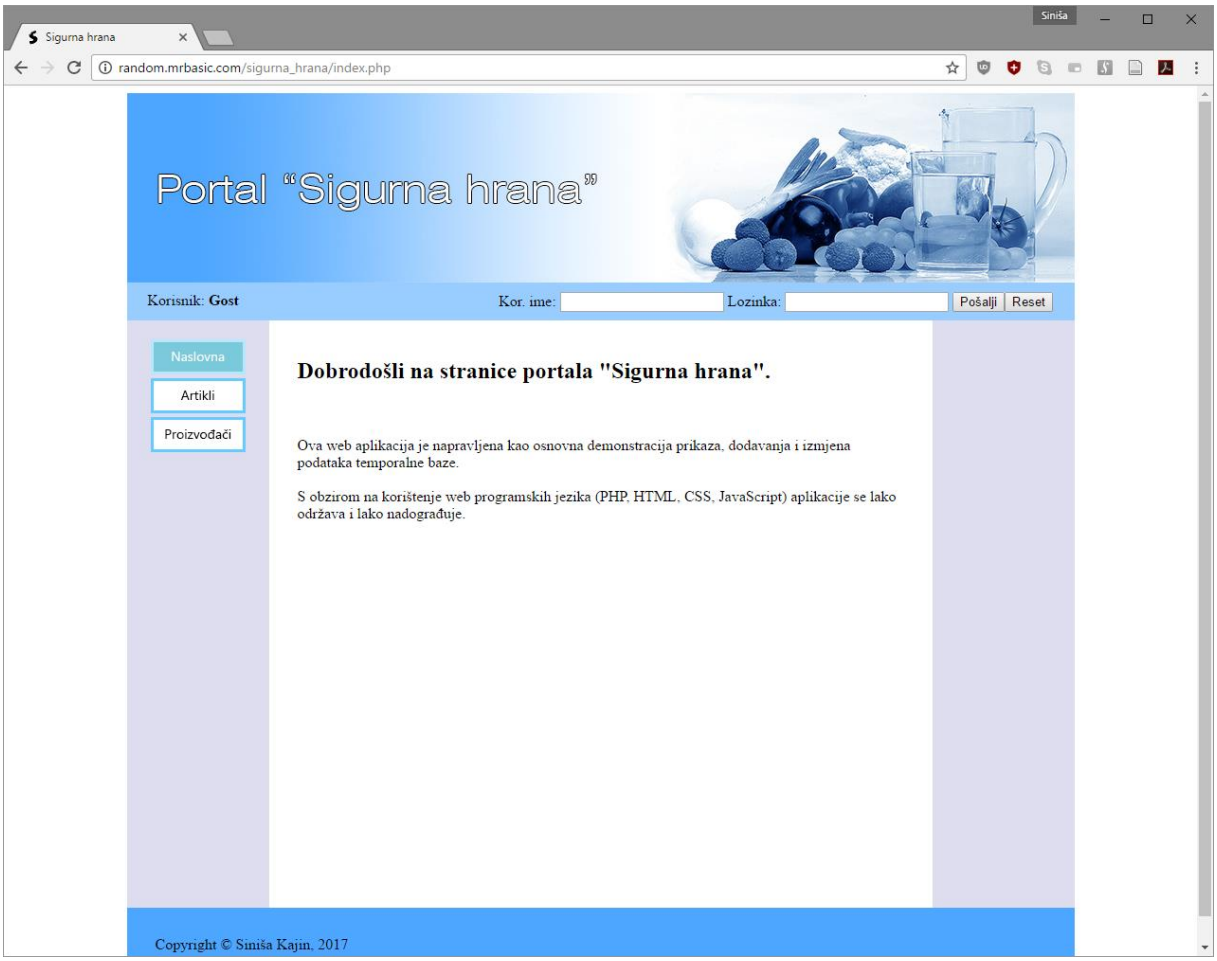
Ako gledamo strukturu ovih tablica, mogao bih ih nazvati bitemporalnim tablicama, iako sama aplikacija za upravljanje bazama podataka ne podržava u potpunosti ovakav tip rada.

Kreiranjem zadnjeg para tablica baza „sigurna_hrana“ spremna je za daljnje korištenje kroz korisničku aplikaciju koju sam izradio nakon akcije kreiranja tablica i okidača.

5.2 Korisnička aplikacija

Kako bih što lakše mogao prikazati rad baze podataka neovisno o platformi na kojoj se korisnička aplikacija pokreće, odlučio sam napraviti web aplikaciju baziranu na PHP programskom jeziku. Tako sam dobio mogućnost izvršavanja aplikacije kroz Internet preglednike na mobilnim platformama i novijim stolnim računalima baziranim na Windows, Linux ili MacOS operativnim sustavima. Aplikacija se lako nadograđuje i održava te je pogodna za prikazivanje koncepta ove temporalne baze podataka.

Početna stranica pruža mogućnost pregleda novosti, prijave korisnika ili pregleda svih artikala i proizvođača u bazi podataka.

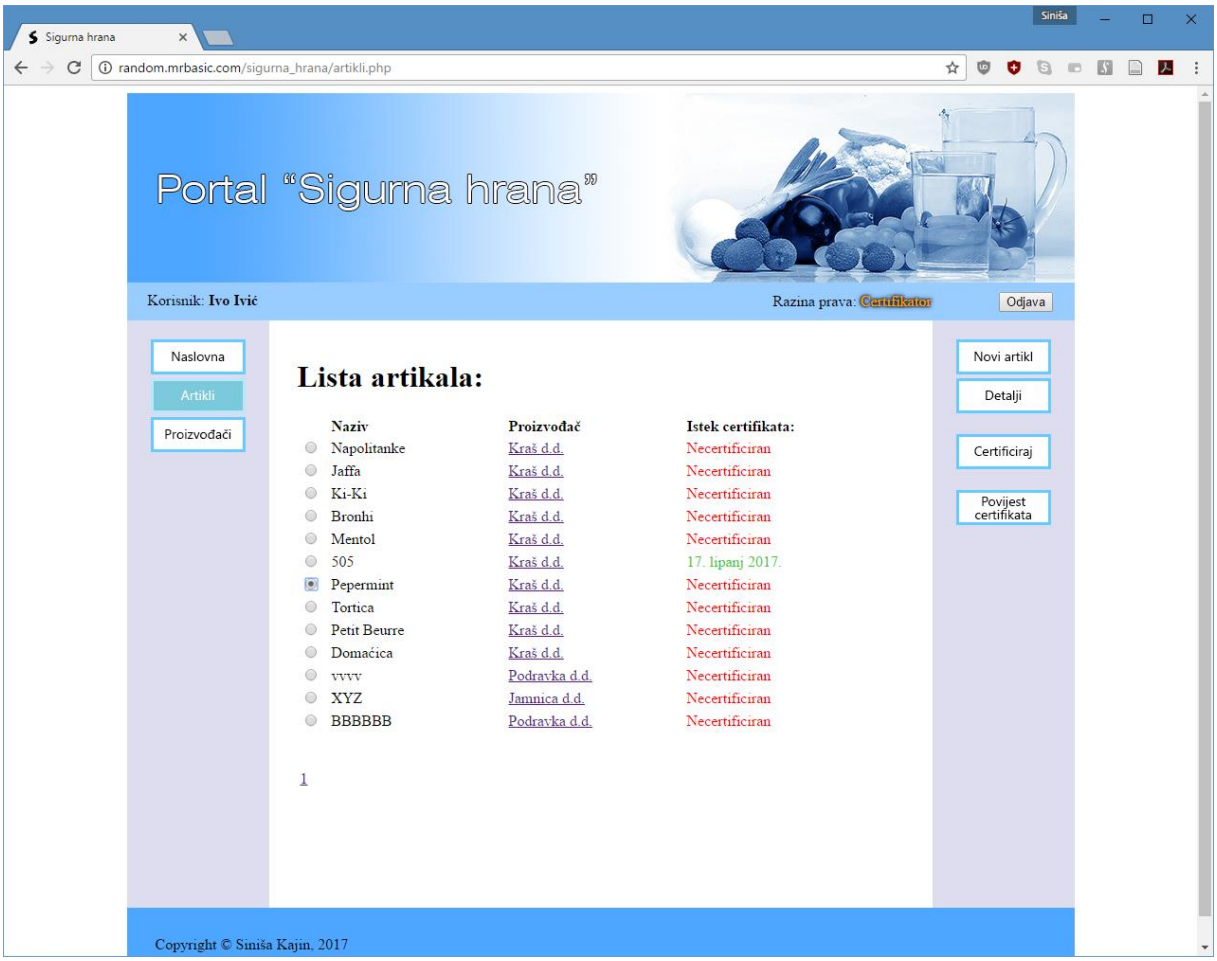


Slika 6. Početna stranica web aplikacije "Sigurna hrana"

Aplikacija pruža različite mogućnosti ovisno o pravima koje korisnik ima. Postavljena su četiri osnovna prava: gost, korisnik, certifikator i administrator. Početno pravo koje ima svaki korisnik kojeg administrator nije registrirao je „Gost“. Gost ima pravo pregleda artikala, detalja artikala, povijesti certifikata, kao i popisa i detalja proizvođača.

Ako je korisnik registriran u bazi podataka, može se prijaviti u sustav pomoću korisničkog imena (koje je u ovom slučaju e-mail adresa) i pripadajuće mu lozinke, i tako stječe pravo pod nazivom „Korisnik“. Mogućnosti su iste kao i kod „Gosta“, s tim da se ovdje pruža mogućnost da korisnik – kad za to stekne uvjete – postane certifikator artikala.

Korisnici sa razinom prava „Certifikator“ imaju veći obim mogućnosti. Samo korisnici koji su prethodno registrirani u bazi podataka sa pravom „Korisnik“ mogu postati certifikatori.



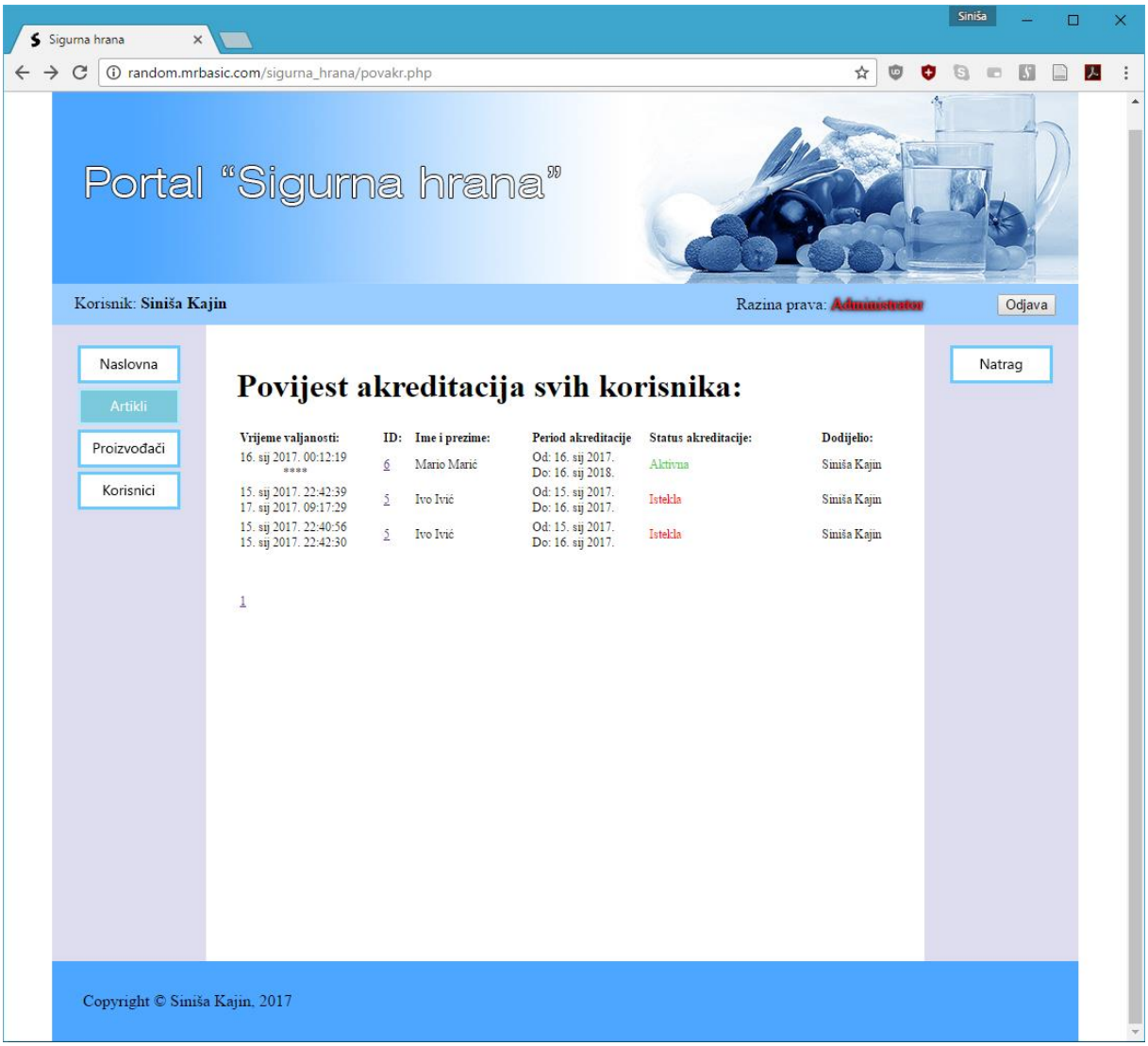
Slika 7. Prikaz mogućnosti korisnika sa razinom prava "Certifikator" nad artiklima.

Nove mogućnosti su dodavanje artikala i njihovog sastava, podataka proizvođača, dodavanje ili promjena naziva gradova koji se koriste u posebnu tablicu u bazi podataka, i u konačnici certificiranje artikla. Kod sastava artikala, aplikacija automatski ažurira popis u slučaju da se neki od sastojaka briše ili mijenja. Certifikator mora unijeti početak i kraj perioda u kojem je artikl certificiran i mora obratiti pozornost na datume koje unosi. U slučaju da se kraj perioda certificiranosti ne unese, aplikacija sama automatski postavlja završni datum točno godinu dana od početnog datuma. U slučaju da je neki od datuma krivo unesen, aplikacija onemogućava potvrđivanje unosa i označava polja u kojima je greška crvenim obrubom. Ako je podatak točno unesen, obrub postaje zelen i nakon što su svi elementi korektno upisani,

omogućava se potvrda unosa. Nakon toga na listi artikala i u detaljima artikla vidimo do kad certifikat traje i tko ga je izdao.

Zadnji tip korisnika je „Administrator“. Bitna informacija je da korisnik koji ima pravo administriranja ne može certificirati artikle, već tu mogućnost ima samo korisnik sa pravom „Certifikator“. Administrator ima mogućnost dodavanja novih korisnika, postavljanje korisnika za certifikatora, pregled povijesti promjena podataka korisnika ili povijest akreditacija korisnika. Šifra korisnika administratora upisuje se u bazu podataka tako da se točno može pratiti koji je administrator dodavao akreditacije certifikatoru.

Kontrola valjanosti akreditacija obavlja se automatski kroz aplikaciju.



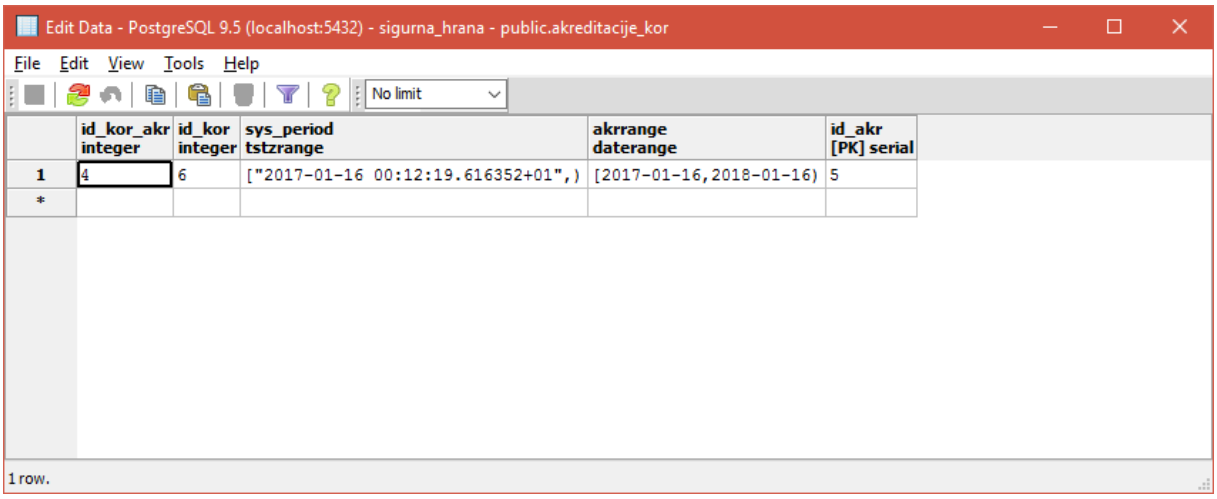
Slika 8. Prikaz povijesti akreditacija korisnika

Kako se za ažurnost baze podataka po pitanju isteka perioda bavi aplikacija, SQL upiti koji kontinuirano ažuriraju bazu podataka se izvršavaju prilikom bilo kojeg upita prema bazi podataka.

```
12  function azuriranje (){
13      global $baza;
14      $azuriranje1 = "DELETE FROM akreditacije_kor WHERE CURRENT_DATE - upper(akrrange) > 0";
15      $azuriranje2 = "UPDATE korisnici SET tip = 2 FROM (SELECT kor.id_kor FROM korisnici kor LEFT JOIN
        akreditacije_kor akr ON akr.id_kor = kor.id_kor WHERE akr.akrrange IS NULL AND kor.tip = 1) as subq WHERE
        korisnici.id_kor = subq.id_kor";
16      $azuriranje3 = "DELETE FROM certif_art WHERE CURRENT_DATE - upper(certrange) > 0";
17      pg_query($baza, $azuriranje1);
18      pg_query($baza, $azuriranje2);
19      pg_query($baza, $azuriranje3);
20  }
21
22  function sviProizvodjaci() {
23      azuriranje();
```

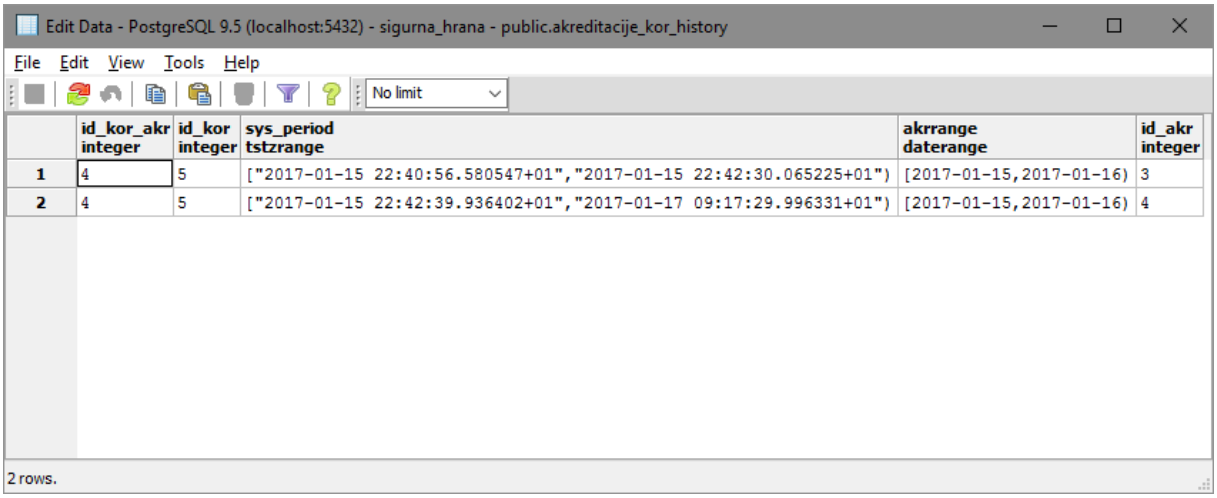
Slika 9. SQL upiti aplikacije kojom se ažuriraju podaci i prikaz poziva funkcije „azuriranje“ iz druge funkcije.

Izvršavaju se tri upita: prvi upit briše iz tablice „akreditacije_kor“ sve entitete kojima je završni datum perioda akreditacije veći od datuma kad se izvršava upit. Unos koji će se brisati prvo se kopira u tablicu „akreditacije_kor_history“ uz unos drugog podatka (trenutak do kad je unos bio valjan) u atribut „sys_period“.



Slika 10. Prikaz unesenih podataka u tablicu "akreditacije_kor".

Na Slici 10. jasno se vidi kako završni datum atributa „sys_period“ ne postoji, što znači da uneseni podatak vrijedi beskonačno, točnije dok se ne dogodi nekakva promjena, bilo da je ručno obavljena ili automatski kroz aplikaciju.



	id_kor_akr integer	id_kor integer	sys_period tstzrange	akrrange daterange	id_akr integer
1	4	5	["2017-01-15 22:40:56.580547+01","2017-01-15 22:42:30.065225+01")	[2017-01-15,2017-01-16)	3
2	4	5	["2017-01-15 22:42:39.936402+01","2017-01-17 09:17:29.996331+01")	[2017-01-15,2017-01-16)	4

Slika 11. Prikaz tablice "akreditacije_kor_history" u kojoj se vide sve izmjene i brisanja u tablici "akreditacije_kor". Atribut „sys_period“ vidljivo ima završni element perioda.

S obzirom na to da se u tablici „korisnici“ svi korisnici koji imaju pravo certificiranja označavaju tipom korisnika 1, moram ažurirati i tu tablicu, pa svim korisnicima koji imaju vrijednost atributa „tip“ 1, a nemaju unos u tablici „akreditacije_kor“, promijeniti vrijednost atributa „tip“ u 2 što znači da taj korisnik prestaje biti certifikator i postaje običan korisnik. Za to koristim drugi upit koji takvo stanje provjerava i mijenja sve korisnike ovisno o navedenim parametrima.

Ovaj dio radnji predstavlja aplikativni temporalni podatak kojeg kontrolira sama aplikacija, dok prijepis podatka u tablicu „akreditacije_kor_history“ sa unosom preostalih podataka predstavlja sistemski temporalni podatak. Iz tog je vidljivo da ovaj set tablica predstavlja bitemporalne tablice.

Na isti način funkcionira i dodjeljivanje certifikata nekom artiklu. Razlika je samo tko ima pravo vršiti takve izmjene (tip korisnika „Certifikator“) i u koje se tablice ti podaci upisuju (tablice „certif_art“ i „certif_art_hist“). Treći upit izvršava se zbog kontrole artikala i njihovih certifikata. U slučaju da je nekom artiklu istekao certifikat, treći upit briše podatke iz tablice „certif_art“ dok kod te izmjene okidač postavljen nad tim podacima vrši kopiranje podataka u tablicu „certif_art_hist“. Ovi podaci su vidljivi svim korisnicima zbog toga što je cilj aplikacije

kontrola certificiranosti nekog artikla na određeni dan što ću kasnije prikazati kroz drugu funkcionalnost aplikacije.

Tablice „korisnici“ i „korisnici_hist“ ne koriste aplikativne temporalne podatke već samo sistemske temporalne podatke, pa tako svi podaci u tablici „korisnici“ nemaju istek roka valjanosti, nego vrijede dok se ne izvrši neka promjena. Tu tablicu ažuriraju isključivo administratori ili SQL upit iz prethodnog primjera koji mijenja tip korisnika iz 1 u 2. Tu je vidljivo kako događaji koji su rezultat podatka iz jedne temporalne tablice utječu na aktivnosti i vrijednosti druge temporalne tablice.

Zadnji aspekt tablice je pregled statusa certificiranosti na određeni datum. Neki artikl nije certificiran u ovom trenutku, ali je bio certificiran nekad u prošlosti. Kao primjer odabrati ću artikl „Bronhi“.

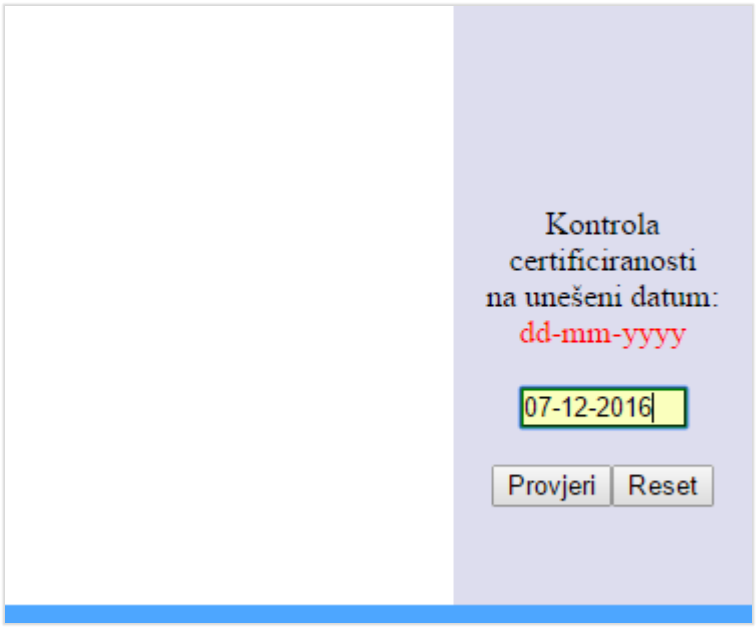
Slika 12. Primjer detalja artikla "Bronhi" koji nije certificiran

Iz slike 12. vidljivo je da u trenutku upita artikl „Bronhi“ nije certificiran. No možda je taj artikl u određenom trenutku bio certificiran. To možemo provjeriti na dva načina. Jedan od načina je provjeriti podatke na stranici „povijest certifikata“.

Slika 13. Detalji artikla na stranici "Povijest certifikata".

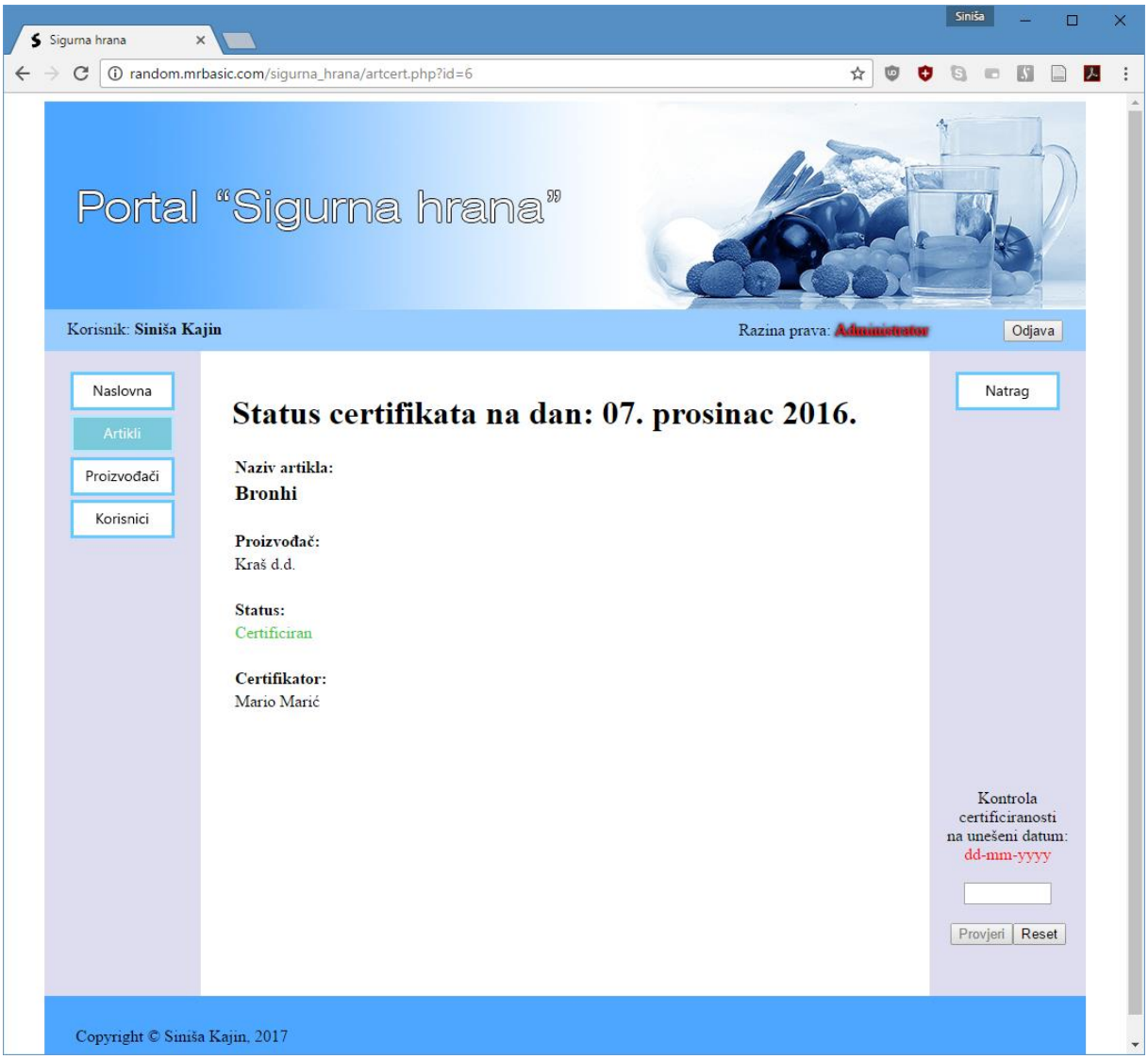
Na ovoj stranici vidi se do kad je certifikat za određeni artikl bio izdan, ali bitna je informacija do kad je taj podatak bio važeći, točnije do kad je taj podatak bio aktualan, jer je možda u nekom trenutku došlo do izmjena i artikl više nije bio certificiran bez obzira na datum koji je naveden u periodu certificiranosti. To vidimo iz podatka u kojem je periodu informacija o certificiranosti bila unesena u tablici za pohranu certifikata i kada se dogodila izmjena.

Drugi način je korištenjem funkcionalnosti provjere certificiranosti za specifičan dan. Na istoj stranici web aplikacije u donjem desnom uglu moguće je unijeti datum i provjeriti je li artikl bio certificiran točno na taj određeni dan.



Slika 14. Funkcionalnost za provjeru certificiranosti artikla na određeni datum.

Kao i kod unosa početka i kraja perioda certificiranosti, skripta koja se izvršava prilikom unosa datuma provjerava je li datum ispravno formatiran i ukoliko jest, prikazuje to zelenom bojom okvira i omogućavanjem pritiska na gumb „Provjeri“. Nakon pritiska na gumb otvara se stranica sa rezultatom pretrage.



Slika 15. Prozor web aplikacije sa rezultatom pretrage certifikata na određeni dan.

Stranica prikazuje naslov sa informacijom na koji datum provjeravamo status certificiranosti te podatke o artiklu, kao i sam status certificiranosti. U slučaju da je artikl na zadani datum bio certificiran, prikazati će se zelenim slovima informacija „Certificiran“ te ispod naziv osobe

koja je certificirala artikl. Ako artikl na taj dan nije bio certificiran, status će se prikazati crveno obojano „Necertificiran“. Za ovakvu kontrolu podataka postoji SQL:2011 naredba s kojom bi se moglo lako doći do rezultata – „AS OF SYSTEM TIME“, te bih ju mogao iskoristiti kako bi došao do potrebnih podataka.

U ovom slučaju SQL upit bi izgledao ovako:

```
sigurna_hrana=# SELECT * FROM akreditacije_kor AS OF SYSTEM TIME
'2016-12-07' WHERE id_art = 6;
```

Na žalost naredbe uvedene u SQL:2011 standardu nisu podržane u ovoj inačici PostgreSQL aplikacije niti kroz proširenje „temporal_tables“ tako da sam morao koristiti operator „&&“ koji provjerava preklapa li se zadana vrijednost sa podatkom unesenim u tablici. U slučaju ove aplikacije SQL upit bi izgledao na slijedeći način:

```
sigurna_hrana=# SELECT id_artikl, certrange, sys_period, sve.id_kor,
korisnici.ime ime, korisnici.prezime prezime FROM (SELECT * FROM
certif_art UNION ALL SELECT * FROM certif_art_hist) AS sve LEFT JOIN
korisnici ON korisnici.id_kor = sve.id_kor WHERE id_artikl = 6 AND
'[2016-12-07 00:00:00,2016-12-07 23:59:59]' && sve.sys_period AND
'[2016-12-07,2016-12-07]' && sve.certrange;
```

Namjerno sam izostavio PHP programski kod jer se u ovom slučaju koristi isključivo kao sredstvo za izvršavanje SQL upita. Pomoću ovako formatiranog upita moguće je dobiti podatke na način da se datum koji smo odabrali (u ovom slučaju to je 07. prosinac 2016.) formatira na razini jednog perioda u trajanju od jednog dana (od 0:00 do 23:59) i da se provjeri preklapa li se takav period sa bilo kojim periodom koji postoji u tablicama „certif_art“ i „certif_art_hist“, uz dodavanje imena i prezimena certifikatora temeljem upisane šifre i uz selekciju samo onih artikala koji imaju identifikacijsku oznaku broj 8. Važno je napomenuti da se spomenuti datum provjerava na način postoji li presjek s datumom certificiranosti (atribut „certrange“) i presjek s datumom unosa u tablicu (atribut „sys_period“).

6. Zaključak

Bez obzira što se ideja o temporalnim bazama podataka razvila 90-ih godina prošlog stoljeća, efektivnog mehanizma i standardiziranog sistema za takav tip podataka u aplikacijama koje upravljaju bazama podataka nije bilo skoro dvadeset godina, čak i nakon što je objavljen prošireni ISO standard 2011. godine koji jasno definira upravljanje temporalnim podacima. Kroz svoj primjer temporalne baze podataka i pripadajuće aplikacije shvatio sam da je ovakav tip baza podataka pogodan za pohranu podataka koji se ne ažuriraju intenzivno već kroz duže periode. U protivnom, količina pohranjenih podataka mogla bi strahovito narasti i dodatno opterećivati sam rad nad takvom bazom podataka. Bez obzira na postavljeni standard, aplikacija se oslanja na rješenja koja ne moraju nužno imati korisničku podršku i time se dodatno destimulira primjena. Naredbe koje su standardizirane nisu uvedene tako da se korisnici moraju oslanjati na kompleksnije setove naredbi i rješenja koja najčešće primjenjuje samo jedan proizvođač aplikacija za upravljanje bazama podataka. Svejedno, novije verzije takvih aplikacija kao što su Microsoft SQL Server 2016 počinju lagano uključivati ovakav tip baza podataka ili tablica u rad iako podrška za standardizirane naredbe i dalje nije uključena.

Na žalost, ako proizvođači ne promijene ovakav tip pristupa prema temporalnim bazama podataka, vjerujem da neće doći do šireg korištenja ovakvih baza podataka i da će se rješenja koja postoje i dalje oslanjati na neka korisnička polu-funkcionalna proširenja. Ovakve baze podataka mogle bi imati odličnu primjenu gdje su potrebne informacije o određenom stanju podataka u određenom trenutku, bilo iz sigurnosnih ili povijesnih razloga.

7. Literatura

1. Bayer, R., *Symmetric binary B-Trees: Data structure and maintenance algorithms - Acta Informatica Vol. 1. Iss. 4.* Springer Berlin Heidelberg, 1972.
2. IBM, *Reference manual – IBM Magnetic tape units, Form A22-6589-1,* IBM, 1962.
3. S. Abiteboul, R. Hull, V. Vianu. *Foundations of Databases,* Addison-Wesley, 1995.
4. Snodgrass R, Ahn I, *Temporal databases - Computer science press,* Rockville, 1986.
5. Codd, E. F., *A relational model of data for large shared data banks,* Communications of the ACM, 1970.
6. Chen, P. P. *The entity-relationship model – toward a unified view of data,* MIT, 1976.
7. Kline, K., Kline D., *SQL in a Nutshell,* O'Reilly & Associates, 2001.
8. ISO/IEC 9075:1992: *Information technology – Database languages – SQL,* International Organization for Standardization, Geneva, Switzerland, 1992.
9. ISO/IEC 9075:2003: *Information technology – Database languages – SQL,* International Organization for Standardization, Geneva, Switzerland, 2003.
10. Allen, J. F., *Maintaining knowledge about temporal intervals,* Communications of the ACM, 1983.
11. ISO/IEC 9075:2008: *Information technology – Database languages – SQL,* International Organization for Standardization, Geneva, Switzerland, 2008.
12. Snodgrass, R., i dr, *TSQL2 Language specifications,* University of Arizona, 1994.
13. ISO/IEC 9075:2011: *Information technology – Database languages – SQL,* International Organization for Standardization, Geneva, Switzerland, 2011.
14. PostgreSQL, *Range functions and operators.* Preuzeto 31. siječnja 2017. s https://www.postgresql.org/docs/9.4/static/functions-range.html -

8. Popis slika

9. Popis tablica