**Baza podataka o nazivima brojeva u raznim jezicima**

Nedavno sam napisao seminar u kojemu sam pokušao napraviti računalni model koji će predviđati evoluciju jezika. Taj sam model evaluirao tako što sam napravio JSON (JavaScript Object Notation) tablicu s nazivima brojeva u prajezicima i njihovim potomcima, te sam mjerio koliko on bolje predviđa kako će se nazivi brojeva mijenjati iz prajezika u jezike od kontrolnog modela koji je te nazive brojeva izmjenjivao nasumično (ne pazeći ni na to da rezultat bude izgovorljiv). Moj je model radio 16.7% bolja predviđanja od kontrolnog modela[[1]](#footnote-2).

Ta JSON tablica korisna je samo za jednu svrhu (ili barem mali broj svrha). Da bih to poboljšao, odlučio sam izraditi bazu podataka iz koje se može upitom napraviti ta tablica, ali i neke druge.

Dakle, specifickacija baze podataka jest:

Potrebno je napraviti bazu podataka s nazivima brojeva od 1 do 10 u raznim jezicima. Svaki jezik ima jedinstveno ime, te može pripadati nekoj familiji jezika (indoeuropski, uralski...). Također, sa svakim je jezikom povezano do 10 naziva brojeva. Svaki naziv broja ima svoju vrijednost (cijeli broj od jedan do deset), riječ (kako se kaže) te može, ali ne mora, imati neki komentar (o neočekivanim glasovnim promjenama...). Svaki jezik može imati samo jedan naziv za pojedini broj od jedan do deset. Jedan jezik može drugome jeziku biti roditelj (primjerice, latinski jezik roditelj je španjolskom i talijanskom jeziku), uz uvijet da roditelj i dijete pripadaju istoj familiji jezika.

Entiteti su: *jezik* i *naziv\_broja*.

Veze su: *jezik-naziv\_broja* (kojem jeziku pripada koji naziv broja), te *jezik-jezik* (jezik roditelj – jezik dijete).

Atributi su: *naziv\_jezika*, *riječ*, *vrijednost\_broja*, *komentar* i *familija\_jezika*.

Entitet *jezik* ima atribute: *naziv\_jezika* i *familija\_jezika*.

Entitet *naziv\_broja* ima atribute: *naziv\_jezika*, *riječ*, *vrijednost\_broja* i *komentar*.

Da bih tu bazu podataka napravio, odlučio sam koristiti SQLite 3, danas daleko najkorišteniji sustav za upravljanje bazama podataka (ugrađen je u danas najpopularniji operativni sustav za mobitele Android za korištenje od strane aplikacija kojima treba jednostavna baza podataka, te ga Google Chrome, danas najpopularniji internetski preglednik, koristi za spremanje cookiesa i povijesti pretraživanja), prije svega zato što sam njega na svoje računalo jedino i uspio instalirati. Moćniji programi za izradu baze podataka, kao što su MySQL ili Microsoft Access, mnogo su teži za instalirati na računalo. SQLite 3 je dovoljno skinuti s njegove internetske stranice kao ZIP arhivu s izvršnim datotekama za Windows[[2]](#footnote-3), ekstraktirati tu ZIP arhivu gdje želimo da nam SQLite 3 bude spremljen na disk računala, i već možemo otvoriti i koristiti SQLite 3. Kad bi barem svi računalni programi bili takvi, ako se već ne daju pokrenuti izravno u internetskom pregledniku! Razumijem da takve programe nije uvijek lagano napraviti. Moj simulator PicoBlaze računala (koji me je profesor Ivan Aleksi zamolio da napravim u slučaju da se laboratorijske vježbe iz kolegija Arhitektura računala ne budu mogli održati zbog pandemije) može se pokrenuti u svakom modernom internetskom pregledniku[[3]](#footnote-4), a da se ništa ne skida niti instalira, ali to je pod cijenu da je znatno manje moćan i estetski ugodan za korištenje nego drugi simulatori PicoBlaze računala. Oba compilera za moj programski jezik[[4]](#footnote-5), i onaj koji cilja na WebAssembly i onaj koji cilja na x86 procesore, na žalost, zahtijevaju da korisnik, da bi ih koristio, mora instalirati vanjski asembler, i bilo bi izuzetno komplicirano izmijeniti te compilere da to ne zahtijevaju.

SQLite 3, kako mu samo ime kaže, omogućuje (baš kao i skoro svi današnji programi za izradu baza podataka) da bazu podataka napravite u programskom jeziku SQL. Zapravo, SQLite 3 poznat je po tome što podržava manje značajki SQL-a nego drugi programi za izradu baza podataka, ali da zato prima gramatički neispravne SQL naredbe dokle god mu je značenje jasno (dok drugi programi za izradu baza podataka uglavnom inzistiraju da SQL koji u njih unosimo strogo poštuje standard). Kad sam razmišljao kako da tu bazu podataka uredim da bude što manje ponavljanja, a da ona ipak može sadržavati sve podatke koje želim u nju staviti (dakle, da bude normalizirana), odlučio sam da je bolje da ne pišem tu bazu podataka cijelu u SQL-u, nego da napišem program u programskom jeziku C++ koji će iz tablice napisane u meni pogodnom formatu proizvesti SQL kod. Taj C++ program dostupan je na mojoj web-stranici[[5]](#footnote-6).

Ta se baza podataka sastoji od tri tablice (relacije), koje su u SQL-u, koji je u tom C++ programu višelinijski string, definirane ovako:

CREATE TABLE languages (language\_name varchar(20),

parent\_language varchar(20),

language\_family varchar(20) default 'Other',

PRIMARY KEY (language\_name),

CONSTRAINT FK\_language\_protolanguage FOREIGN KEY

(language\_name) REFERENCES languages(language\_name),

CONSTRAINT valid\_language\_family\_name CHECK (language\_family in

('Niger–Congo', 'Austronesian', 'Sino-Tibetan', 'Indo-European',

'Australian', 'Afro-Asiatic', 'Nilo-Saharan', 'Oto-Manguean',

'Austroasiatic', 'Tai–Kadai', 'Dravidian', 'Tupian', 'Other'))

);

CREATE TABLE names\_of\_numbers (

language\_name varchar(20),

word varchar(20),

value integer,

comment varchar(1000),

attested integer check (attested in (0,1)),

CONSTRAINT FK\_number\_language FOREIGN KEY (language\_name) REFERENCES languages(language\_name),

CONSTRAINT value\_number\_between\_1\_and\_10 CHECK (value between 1 and 10),

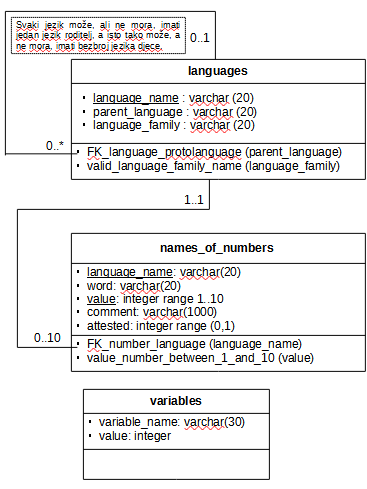
PRIMARY KEY (language\_name, value)

);

CREATE TABLE variables (variable\_name varchar(30) primary key, value integer);

Dakle, te se tablice zovu *languages*, *names\_of\_numbers* i *variables*. Tablica *languages* predstavlja jezike, *names\_of\_numbers* predstavlja nazive brojeva, a *variables* je pomoćna tablica koju koristim za spremanje varijabli da bi upiti bili kraći i čitljiviji. Tablica *variables* ne bi bila potrebna da koristim neki moćniji program za izradu baze podataka, ali SQLite 3, na žalost, ne podržava varijable u SQL kodu.

Dakle, kao UML dijagram, interna strukture te baze podataka mogla bi se prikazati ovako:

Očito, u toj bazi podataka nema mjesta za kreole i pidžine, jezike s dva ili više roditelja. Da to napravimo, morali bismo napraviti još jednu tablicu koja bi predstavljala tu vezu roditelj-dijete, čiji bi se primarni ključ sastojao od dva strana ključa koja referiraju na pojedine jezike (kako se veze više-na-više rade u relacijskim bazama podataka). Smatram da bi to bazu podataka učinilo znatno kompliciranijom, te da nije vrijedno toga.

Nakon toga u tom mom C++ programu slijedi tablica s podacima unesena kao višelinijski string. Jedan od redaka u toj tablici jest:

Italian Indo-European Latin uno due tre kuattro txinkwe sei sette otto nove dietxi

Dakle, u toj se tablici, kako sam unosio podatke, u prvome stupcu nalazi naziv jezika, u ovom slučaju to je talijanski jezik. U drugom stupcu nalazi se naziv obitelji jezika kojoj taj jezik pripada, a talijanski jezik pripada indoeuropskoj skupini jezika. U trećem stupcu nalazi se naziv jezika roditelja, a roditelj talijanskog jezika jest latinski jezik. Redak s podacima o jeziku roditelju mora dolaziti prije retka s podacima o jeziku djetetu, inače će se SQLite buniti da *foreign key constraint* nije zadovoljen.Nakon toga slijede nazivi brojeva od jedan do deset u tom jeziku, ne po pravopisu tog jezika, već po pojednostavljenom sustavu transkripcije koji sam osmislio i primijenio u onoj JSON tablici. U njemu se glas '*š*' piše '*x*', jer što će nam znak koji se može zamijeniti sa „*ks*”, a treba nam još koji znak da označimo glas '*š*'? Glas '*č*' piše se, naravno, „*tx*”.

Riječi koje nisu potvrđene(*attested*), ali se pretpostavlja da su postojale, označavaju se, kao i inače u lingvistici, tako da se ispred njih stavi zvjezdica \*. Ako neki podatak želimo preskočiti, tamo stavimo NULL. Recimo, za afroazijatski prajezik, redak u tablici izgleda ovako:

Proto-Afro-Asiatic Afro-Asiatic NULL NULL \*tsan \*xaimz \*fazu NULL NULL NULL NULL NULL NULL

Afroazijatski prajezik, rekonstruirani jezik koji se govorio u današnjoj Etiopiji oko 12'000 godina prije nove ere (na kraju zadnjeg ledenog doba), naravno, pripadao je afroazijatskoj skupini jezika. To je najraniji prajezik s općeprihvaćenom rekonstrukcijom. Afroazijatski prajezik nije imao poznatog roditelja, i zato je u trećem stupcu NULL. Riječ za broj jedan nije rekonstruirana, jer je ona, iako jedna od najstabilnijih riječi u jezicima, ipak manje stabilna od naziva za brojeve od dva do deset, i ponekad se zna zamijeniti za riječi sa značenjima *sam* ili *prst*, a u desetak tisuća godina koliko je prošlo od prajezika do najranijeg zapisa tog jezika (prvi pisani afroazijatski jezik bio je staroegipatski, a oko petsto godina nakon toga imamo najranije tekstove na akadskom jeziku), to se valjda dogodilo u svakom afroazijatskom jeziku. Riječ za broj dva u afroazijatskom prajeziku rekonstruira se kao „*can”*, riječ za broj tri kao „*šaimz*”, riječ za broj četiri kao „*fazu*”, a riječi za broj pet i više nisu rekonstruirane, vjerojatno zato što ih jezik koji se govorio prije 14'000 godina nije ni imao. Naravno, ispred svih tih riječi ide znak \* da su hipotetske, jer govorimo o jeziku koji se govorio desetak tisuća godina prije pojave pisma.

Nakon te tablice slijedi C++ petlja koja je pretvara u SQL:

istringstream tableStream(table);

while (tableStream) {

string languageName, languageFamilyName, parentLanguageName;

tableStream >> languageName >> languageFamilyName >> parentLanguageName;

if (!tableStream)

break;

result += R"(

INSERT INTO languages values (')" +

languageName + "'," +

(parentLanguageName != "NULL" ? "'" + parentLanguageName + "'"

: "NULL") +

",'" + languageFamilyName + "');\n";

for (int i = 1; i <= 10; i++) {

string numberName;

tableStream >> numberName;

if (numberName == "NULL")

continue;

result += "INSERT INTO names\_of\_numbers(language\_name, word, value, "

"attested) VALUES ('" +

languageName + "','" +

(numberName[0] == '\*' ? numberName.substr(1) : numberName) +

"'," + to\_string(i) + "," + (numberName[0] == '\*' ? "0" : "1") +

");\n";

}

}

Recimo, on redak o afroazijatskom prajeziku pretvara u ovakav SQL:

INSERT INTO languages values ('Proto-Afro-Asiatic',NULL,'Afro-Asiatic');

INSERT INTO names\_of\_numbers(language\_name, word, value, attested) VALUES ('Proto-Afro-Asiatic','tsan',2,0);

INSERT INTO names\_of\_numbers(language\_name, word, value, attested) VALUES ('Proto-Afro-Asiatic','xaimz',3,0);

INSERT INTO names\_of\_numbers(language\_name, word, value, attested) VALUES ('Proto-Afro-Asiatic','fazu',4,0);

Nakon toga slijedi jedan upit u SQL kodu koji je višelinijski string u C++ kodu, koji provjerava da nismo slučajno naveli da roditelj nekog jezika pripada drugoj familiji jezika (afroazijatski jezik, recimo, ne može biti roditelj indoeuropskom jeziku):

SELECT language1.language\_name AS

"Languages with different language family from their parent language, check for these errors!"

FROM (languages AS language1),languages WHERE

language1.language\_name=languages.parent\_language and

language1.language\_family<>languages.language\_family;

Ovo je potrebno jer SQLite, za razliku od naprednijih programa za izradu baza podataka, ne podržava ograničenja (*constraints*) koja se odnose na vanjske tablice, to jest, u SQLiteu ograničenja se mogu odnosi isključivo ili na strane ključeve ili na vrijednosti unutar tablice. Da radimo u nekom naprednijem programu za izradu baza podataka, mogli bismo napraviti ograničenje u CREATE TABLEkoje bi nas upozorilo čim pokušamo INSERT-ati redak koji takvu grešku sadrži.

Nakon toga slijedi niz naredbi koje će na pojedine riječi naredbom UPDATE umetnuti komentare te će naredbom SELECT provjeriti je li to dobro napravljeno:

UPDATE names\_of\_numbers SET comment='It is amazing to me that the Proto-Sino-Tibetan word for the number 8 was "triat", when, in nearly all modern Sino-Tibetan languages, the word for number eight starts with either "p" or "b".' WHERE language\_name='Proto-Sino-Tibetan' and value=8;

UPDATE names\_of\_numbers SET comment='The Etruscan word for the number 8, "kezp", is usually considered to be compound from "ki" (three) and "zep" (hand), to mean "three plus the number of fingers on one hand".' WHERE language\_name='Etruscan' and value=8;

UPDATE names\_of\_numbers SET comment='Some amazing sound changes visible here...' WHERE language\_name='Armenian' and value=2;

UPDATE names\_of\_numbers, languages SET comment='Innovation in the Anatolian branch...' WHERE value=4 and word like 'm%' and languages.language\_name=names\_of\_numbers.language\_name and languages.language\_family='Indo-European';

UPDATE names\_of\_numbers, languages SET comment='The "d" at the beginning is either contamination from the word for the number 10, or sound change from "hn".' WHERE value=9 and languages.language\_name=names\_of\_numbers.language\_name and languages.language\_family='Indo-European' and word like 'd%';

SELECT \* FROM names\_of\_numbers WHERE comment IS NOT NULL;

SQLite jedan je od rijetkih programa za izradu baza podataka koji dopušta da se u naredbi UPDATE referencira više tablica odjednom, jer to nije po standardu. To mu je jedna od prednosti, kao i to da se lagano instalira.

Nakon toga slijede deklaracije *varijabli* da se skrate upiti i da budu čitljiviji:

INSERT INTO variables VALUES ('Number of languages',(SELECT COUNT(\*) FROM Languages));

INSERT INTO variables VALUES ('Number of Indo-European languages',(SELECT COUNT(\*) FROM Languages WHERE Languages.language\_family='Indo-European'));

INSERT INTO variables VALUES ('Number of Semitic languages',(SELECT COUNT(\*) FROM Languages WHERE Languages.parent\_language='Proto-Semitic'));

Iako te *varijable* ne čine upite onoliko kratkima i čitljivima kako ih čine prave varijable koje postoje u naprednijim programima za izradu baza podataka.

Onda slijedi upit o tome u koliko posto jezika riječi za brojeve *dva* i *deset* počinju istim slovom, a vjerojatno je u sličnom postotku jezika *deset* dolazi nekada u povijesti od složenice koja je značila *dvije šake*.

SELECT 100.\*COUNT(DISTINCT names\_of\_numbers.language\_name)/variables.value

AS "Percentage of languages in which the words for 2 and 10 start with the same letter."

FROM (names\_of\_numbers as numbers1),names\_of\_numbers,variables

WHERE variables.variable\_name='Number of languages' and numbers1.language\_name=names\_of\_numbers.language\_name and substr(numbers1.word,1,1)=substr(names\_of\_numbers.word,1,1) and numbers1.value=2 and names\_of\_numbers.value=10;

Na taj upit SQLite odgovara sa:

29.6052631578947

Jedan od takvih jezika bio je, pretpostavlja se, indoeuropski prajezik. Pa, hajdemo pitati SQLite za koliki postotak indoeuropskih jezika to vrijedi:

SELECT 100.\*COUNT(DISTINCT names\_of\_numbers.language\_name)/variables.value

AS "Percentage of Indo-European languages in which the words for 2 and 10 start with the same letter."

FROM (names\_of\_numbers as numbers1),names\_of\_numbers,languages,variables

WHERE variables.variable\_name='Number of Indo-European languages' and numbers1.language\_name=names\_of\_numbers.language\_name

and names\_of\_numbers.language\_name=languages.language\_name and languages.language\_family='Indo-European' and

substr(numbers1.word,1,1)=substr(names\_of\_numbers.word,1,1) and numbers1.value=2 and names\_of\_numbers.value=10;

SQLite na to odgovara sa:

78.7234042553192

Dakle, u nekih 20-ak posto indoeuropskih jezika dogodila se glasovna promjena da brojevi *dva* i *deset* više ne počinju istim slovom. Najpoznatiji takav jezik svakako je armenski, gdje je riječ za broj deset *tasun*, a riječ za broj dva je *erku*. U armenskom je *du* na početku riječi prelaziko u *erk*.Isto tako je armenska riječ za *dugovječan*, od korijena odakle dolazi latinska riječ *durus*, *erkar*.

Na etrurskom jeziku riječ za broj osam, *kezp*, složenica je od riječi za broj tri, *ki*, i riječi za ruku, *zep*, dakle, znači „*tri plus prsti na ruci*”. Koliko često riječ za broj osam nastaje tako? Pa, upitajmo u koliko jezika riječ za broj tri i riječ za broj osam počinje istim slovom.

SELECT 100. \* COUNT (DISTINCT names\_of\_numbers.language\_name) / (SELECT value FROM variables WHERE variable\_name = 'Number of languages')

AS "Percentage of languages in which the words for three and eight start with the same letter (as in Etruscan, where 'eight' means 'three plus hand')."

FROM (names\_of\_numbers as numbers1), names\_of\_numbers

WHERE numbers1.language\_name=names\_of\_numbers.language\_name and

substr(numbers1.word,1,1)=substr(names\_of\_numbers.word,1,1) and

numbers1.value=3 and names\_of\_numbers.value=8;

SQLite na taj upit odgovara sa:

17.1052631578947

Tko zna, možda je neki od tih 17% jezika uistinu daleki srodnik etrurskog jezika. Iako je mala vjerojatnost, jer se složenica *kezp* čini relativno prozirna (riječ za tri i riječ za ruku posve su prepoznatljive u njoj), a ne kao da je nastala u nekom davnom prajeziku.

A u koliko jezika vrijedi suprotno nego u etrurskome, da *osam* ne dolazi od „*tri plus pet”* nego od „*dva do deset”*? Upitajmo u koliko jezika riječ za osam počinje s istim slovom kao riječ za broj dva.

SELECT 100. \* COUNT (DISTINCT names\_of\_numbers.language\_name) / (SELECT value FROM variables WHERE variable\_name = 'Number of languages')

AS "Percentage of languages in which the words for two and eight start with the same letter (the opposite of Etruscan, so that 'eight' is 'two to ten')."

FROM (names\_of\_numbers as numbers1), names\_of\_numbers

WHERE numbers1.language\_name=names\_of\_numbers.language\_name and

substr(numbers1.word,1,1)=substr(names\_of\_numbers.word,1,1) and

numbers1.value=2 and names\_of\_numbers.value=8;

SQLite je na taj upit odgovorio sa:

11.8421052631579

Dakle, riječ za broj osam 17.1/11.8-1=45% češće dolazi od složenice sa značenjem „*tri plus pet*” nego od složenice sa značenjem „*dva do deset*”. Ima donekle smisla: logično je da prvo u jeziku nastane riječ za broj osam, a tek onda riječ za broj deset (iako ima izuzetaka, recimo, uralski prajezik imao je riječ za broj deset, ali ne i riječ za broj osam). U indoeuropskom prajeziku riječ za broj osam, \**h3ekjtow*, nije prozirne etimologije. Neki misle da se to treba čitati kao *dva puta četiri*, da je to arhajska riječ za četiri (odakle dolazi avestinska riječ *ašti* koja je označavala prste na ruci osim palca)plus nastavak za dual, ali mislim da ta etimologija baš ne pripada glavnoj struji lingvistike.

Upitajmo zatim u koliko posto jezika riječi za *šest* i *sedam* počinju s istim slovom, što iz nekog meni nedokučivog razloga očito vrijedi za velik broj jezika:

SELECT 100.\*COUNT(DISTINCT names\_of\_numbers.language\_name)/variables.value

AS "Percentage of languages in which the words for 6 and 7 start with the same letter."

FROM (names\_of\_numbers as numbers1),names\_of\_numbers,variables

WHERE variables.variable\_name='Number of languages' and numbers1.language\_name=names\_of\_numbers.language\_name and substr(numbers1.word,1,1)=substr(names\_of\_numbers.word,1,1) and numbers1.value=6 and names\_of\_numbers.value=7;

SQLite na to odgovara sa:

38.1578947368421

Isto je, pretpostavlja se, vrijedilo i za semitski prajezik (potomak afroazijatskog prajezika, koji se govorio oko 3'000 godina prije nove ere, u isto vrijeme kad i indoeuropski prajezik i sumeranski jezik) i za kasne stadije indoeuropskog prajezika. Pa u kolikom postotku indoeuropskih jezika riječi za šest i sedam počinju istim slovom?

SELECT 100.\*COUNT(DISTINCT names\_of\_numbers.language\_name)/variables.value

AS "Percentage of Indo-European languages in which the words for 6 and 7 start with the same letter."

FROM (names\_of\_numbers as numbers1),names\_of\_numbers,languages,variables

WHERE variables.variable\_name='Number of Indo-European languages' and numbers1.language\_name=names\_of\_numbers.language\_name

and names\_of\_numbers.language\_name=languages.language\_name and languages.language\_family='Indo-European' and

substr(numbers1.word,1,1)=substr(names\_of\_numbers.word,1,1) and numbers1.value=6 and names\_of\_numbers.value=7;

SQLite na to odgovara sa:

59.5744680851064

Očito se u mnogim jezicima, kao i u hrvatskom, dogodila neka vrsta palatalizacije ili u riječi za broj šest, ili u riječi za broj sedam, ali ne i oboje. Isto tako, neki misle da u ranom indoeuropskom, prije no što su se od indoeuropskog prajezika odvojili preci anatolijskih jezika (hetitski i njemu srodni) i armenskog jezika, riječ za broj šest nije bila *\*swekjs*, nego da je bila *\*wekjs*, a da je početno '*s*' dodano u riječ za broj šest zbog analogije s riječju za broj sedam.

A kako je u semitskim jezicima?

SELECT 100.\*COUNT(DISTINCT names\_of\_numbers.language\_name)/variables.value

AS "Percentage of Semitic languages in which the words for 6 and 7 start with the same letter."

FROM (names\_of\_numbers as numbers1),names\_of\_numbers,languages,variables

WHERE variables.variable\_name='Number of Semitic languages' and numbers1.language\_name=names\_of\_numbers.language\_name

and names\_of\_numbers.language\_name=languages.language\_name and languages.parent\_language='Proto-Semitic' and

substr(numbers1.word,1,1)=substr(names\_of\_numbers.word,1,1) and numbers1.value=6 and names\_of\_numbers.value=7;

SQLite na to odgovara da to vrijedi za sve semitske jezike koje sam unio u bazu podataka:

100.0

Zapanjilo kad sam bio doznao da se riječ za broj osam u sinotibetanskom prajeziku rekonstruira kao *\*triat*, kad znam da u većini sinotibetanskih jezika riječ za broj osam počinje ili sa *p* ili sa *b*. U koliko posto jezika riječ za broj osam počinje sa *p* ili sa *b*?

SELECT 100.\*COUNT(DISTINCT names\_of\_numbers.language\_name)/(SELECT value FROM variables WHERE variable\_name='Number of languages')

AS "Percentage of languages in which the word for the number 8 starts with either 'p' or 'b'."

FROM languages,names\_of\_numbers

WHERE names\_of\_numbers.value=8 and (names\_of\_numbers.word like 'p%' or names\_of\_numbers.word like 'b%') and names\_of\_numbers.language\_name=languages.language\_name;

SQLite na to odgovara sa:

13.1578947368421

A za koliki postotak sinotibetanskih jezika to vrijedi?

SELECT 100.\*COUNT(DISTINCT names\_of\_numbers.language\_name)/(SELECT COUNT(\*) FROM languages WHERE language\_family='Sino-Tibetan')

AS "Percentage of Sino-Tibetan languages in which the word for the number 8 starts with either 'p' or 'b'."

FROM languages,names\_of\_numbers

WHERE names\_of\_numbers.value=8 and (names\_of\_numbers.word like 'p%' or names\_of\_numbers.word like 'b%') and names\_of\_numbers.language\_name=languages.language\_name and languages.language\_family='Sino-Tibetan';

SQLite na to odgovara sa:

70.0

Kao što sam napisao u fusnoti u seminaru o fonološkoj evoluciji jezika: „Zapanjujuće je to što su riječi za brojeve *šest* i *sedam* slične u mnogim nesrodnim jezicima: *sedam* se na arapskom kaže *sabha*, a na ostalim semitskim jezicima slično, na latinskom se kaže *septem*, a na drugim indoeuropskim jezicima slično, a na etrurskom se kaže *semf*. Broj *šest* se na latinskom kaže *sex*, a na drugim indoeuropskim jezicima slično, na baskijskom se kaže *sei*, na etrurskom se kaže *sa*, na arapskom jeziku *sitta*, a na ostalim semitskim jezicima slično. S obzirom da nema očitog objašnjenja kako bi svi ti nazivi brojeva mogli biti povezani, vjerojatno je riječ o slučajnosti.”. Pa u koliko jezika riječ za broj šest počinje sa *'š'* ili sa *'s'*?

SELECT 100.\*COUNT(\*)/variables.value

AS "Percentage of languages in which the word for 6 starts with 's' or 'sh' (here transcribed as 'x')."

FROM names\_of\_numbers,variables

WHERE variables.variable\_name='Number of languages' and names\_of\_numbers.value=6 and (names\_of\_numbers.word like 's%' or names\_of\_numbers.word like 'x%');

SQLite na to odgovara sa:

39.4736842105263

A što je s riječima za broj sedam u raznim jezicima?

SELECT 100.\*COUNT(\*)/variables.value

AS "Percentage of languages in which the word for 7 starts with 's' or 'sh'."

FROM names\_of\_numbers,variables

WHERE variables.variable\_name='Number of languages' and names\_of\_numbers.value=7 and (names\_of\_numbers.word like 's%' or names\_of\_numbers.word like 'x%');

SQLite na to odgovara sa:

44.7368421052632

Pa koja je to nekolicina jezika u kojima ili riječ za šest počinje sa '*s*' ili '*š*', ili riječ za sedam počinje sa '*s*' ili '*š*', ali ne i oboje?

SELECT language\_family as "Language family", number1.language\_name as "Language in which exactly one of the words for 6 and 7 seems to be borrowed from the mystery language", number1.word as "Name for number 6", names\_of\_numbers.word as "Name for number 7"

FROM (names\_of\_numbers AS number1),names\_of\_numbers,languages

WHERE number1.language\_name=names\_of\_numbers.language\_name and names\_of\_numbers.language\_name=languages.language\_name

and number1.value=6 and names\_of\_numbers.value=7

and (names\_of\_numbers.word like 's%' or number1.word like 's%' or names\_of\_numbers.word like 'x%' or number1.word like 'x%') and substr(names\_of\_numbers.word,1,1)<>substr(number1.word,1,1) and not(number1.word like 's%' and names\_of\_numbers.word like 'x%') and not(number1.word like 'x%' and names\_of\_numbers.word like 's%');

SQLite na to odgovara sa:

Afro-Asiatic|Zenaga|shodush|ishsha

Afro-Asiatic|Tamasheq|sedis|essa

Afro-Asiatic|Hausa|shida|bakwai

Afro-Asiatic|Late\_Egyptian|hussaw|safh

Afro-Asiatic|Bench|sapm|napm

Afro-Asiatic|Chara|safun|lapun

Indo-European|Albanian|giaxte|xtate

Indo-European|Pali|txa|satta

Indo-European|Hindi|txe|sat

Indo-European|Nepali|chha|sat

Indo-European|Bengali|txoe|xat

Indo-European|Punjabi|khuhe|satut

Indo-European|Sindhi|txah|sath

Sino-Tibetan|Proto-Sino-Tibetan|triuk|snis

Sino-Tibetan|Yi|fu|su

Austroasiatic|Vietnamese|sau|bay

Other|Proto-Uralic|kutte|xajtxem

Other|Finnish|kuusi|seitseman

Other|Estonian|kuus|seitse

Other|Mansi|hot|sat

Other|Georgian|ekusi|xvidi

Other|Quechua|suquuta|quantxis

Other|Basque|sei|zazpi

Drugi programi za pravljanje baza podataka crtaju ljepše tablice nego što ih crta SQLite.

Evo, nadam se da će ovo što sam napisao potaknuti još nekoga da krene istraživati podrijetlo naziva brojeva, možda otkrijemo još nešto zanimljivo.

1. https://flatassembler.github.io/Fonoloska\_evolucija\_jezika.docx [↑](#footnote-ref-2)
2. https://www.sqlite.org/2021/sqlite-tools-win32-x86-3360000.zip [↑](#footnote-ref-3)
3. https://flatassembler.github.io/PicoBlaze/PicoBlaze [↑](#footnote-ref-4)
4. https://flatassembler.github.io/AEC\_specification#HowToCompile [↑](#footnote-ref-5)
5. https://flatassembler.github.io/language\_database.cpp [↑](#footnote-ref-6)