|  |  |  |
| --- | --- | --- |
|  | UNIVERZITET U NIŠU  ELEKTRONSKI FAKULTET |  |

Vladimir Nešić

**Interna struktura i organizacija indeksa kod postgresql-a**

SEMINARSKI RAD

Predmet: Sistemi za upravljanje bazama podataka

|  |
| --- |
| Student: |
|  |
| Vladimir Nešić, br. ind. 1235 |

Niš, april 2021. god.

Contents

[**1.** **INTERNA STRUKTURA POSTGRESQL-a** 4](#_Toc69340732)

[1.1. Logička struktura 4](#_Toc69340733)

[1.2. Fizička struktura 5](#_Toc69340734)

[1.2.2. Layout baza podataka 5](#_Toc69340735)

[1.2.3 Layout datoteka povezanih sa tabelama i indeksima 5](#_Toc69340736)

[1.2.4. Tablespaces 6](#_Toc69340737)

[1.3 Interna struktura Heap Table fajla 7](#_Toc69340738)

[1.4 PostgreSQL system catalog 8](#_Toc69340739)

[1.4.1 Pg\_database 8](#_Toc69340740)

[1.4.2 Pg\_stat\_database 9](#_Toc69340741)

[1.4.3 Pg\_stat\_activity 10](#_Toc69340742)

[1.4.4 Pg\_locks 10](#_Toc69340743)

[1.4.5 Pg\_stat\_user\_tables 11](#_Toc69340744)

[2. ORGANIZACIJA INDEKSA 11](#_Toc69340745)

[2.1 Tipovi indeksa 12](#_Toc69340746)

[2.1.1 Btree (B-stabla) 12](#_Toc69340747)

[2.1.2 Hash 15](#_Toc69340748)

[2.1.3 GiST 16](#_Toc69340749)

[2.2 Parcijalni indeksi 23](#_Toc69340750)

[2.2 Expression indeksi 23](#_Toc69340751)

[2.2 Unique indeksi 23](#_Toc69340752)

[3. LITERATURA 24](#_Toc69340753)

PostgreSQL je objektno-relacioni sistem za upravljanje bazama podataka (objektno-relacioni DBMS ili ORDBMS), razvien na osnovu Berkley-vog sistema za upravljanje bazama podataka Postgres. PostgreSQL sadrži moćan objektno-relacioni model podataka, bogat izbor vrsta podataka, laku nadogradivost, kao i bogat set SQL naredbi.

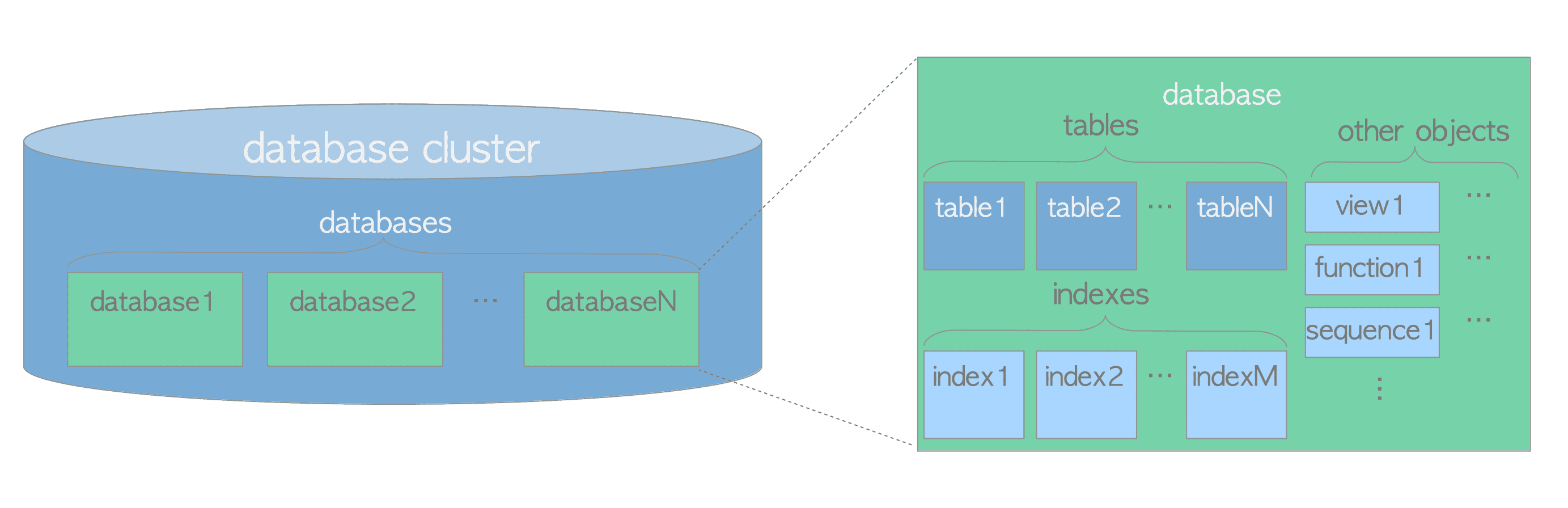
Zbog svoje open-source prirode, PostgreSQL može da koristi, menja i distribuira bilo ko u bilo koju svrhu, bilo privatnu, komercijalnu ili akademsku.

## **INTERNA STRUKTURA POSTGRESQL-a**

## 1.1. Logička struktura

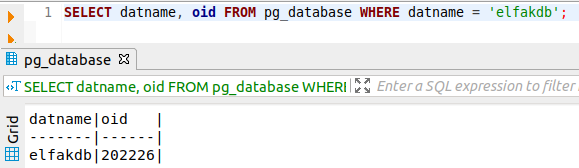
PostgreSQL server radi na jednom hostu i upravlja jednim klasterom baze podataka koji predstavlja skup baza podataka na tom serveru.

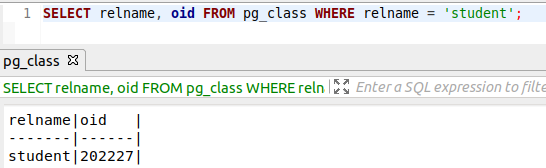
Na slici 1 je prikazana logička struktura klastera baze podataka. Baza podataka je kolekcija objekata baze podataka. U relacionoj teoriji baze podataka, objekat baze podataka je struktura podataka koja se koristi ili za čuvanje ili za referenciranje podataka. Tabela je tipičan primer za čuvanje, dok su za referenciranje zaduženi indeksi, pogledi, funkcije, itd. U PostgreSQL-u, same baze podataka predstavljaju takođe i objekte baze podataka koji su logički odvojeni jedni od drugih. Svi ostali objekti baze podataka (npr. tabele, indeksi itd.) pripadaju njihovim bazama podataka.



*Slika 1. Logička struktura PostgreSQL-a*

Svim objektima baze podataka u PostgreSQL-u se upravlja odgovarajućim identifikatorima objekata (OID). Odnosi između objekata baze podataka i odgovarajućih OID-ova čuvaju se u odgovarajućim sistemskim katalozima, u zavisnosti od vrste objekata. Na primer, identifikatori baza podataka i tabela čuvaju se u pg\_database i pg\_class respektivno (Slika 2):



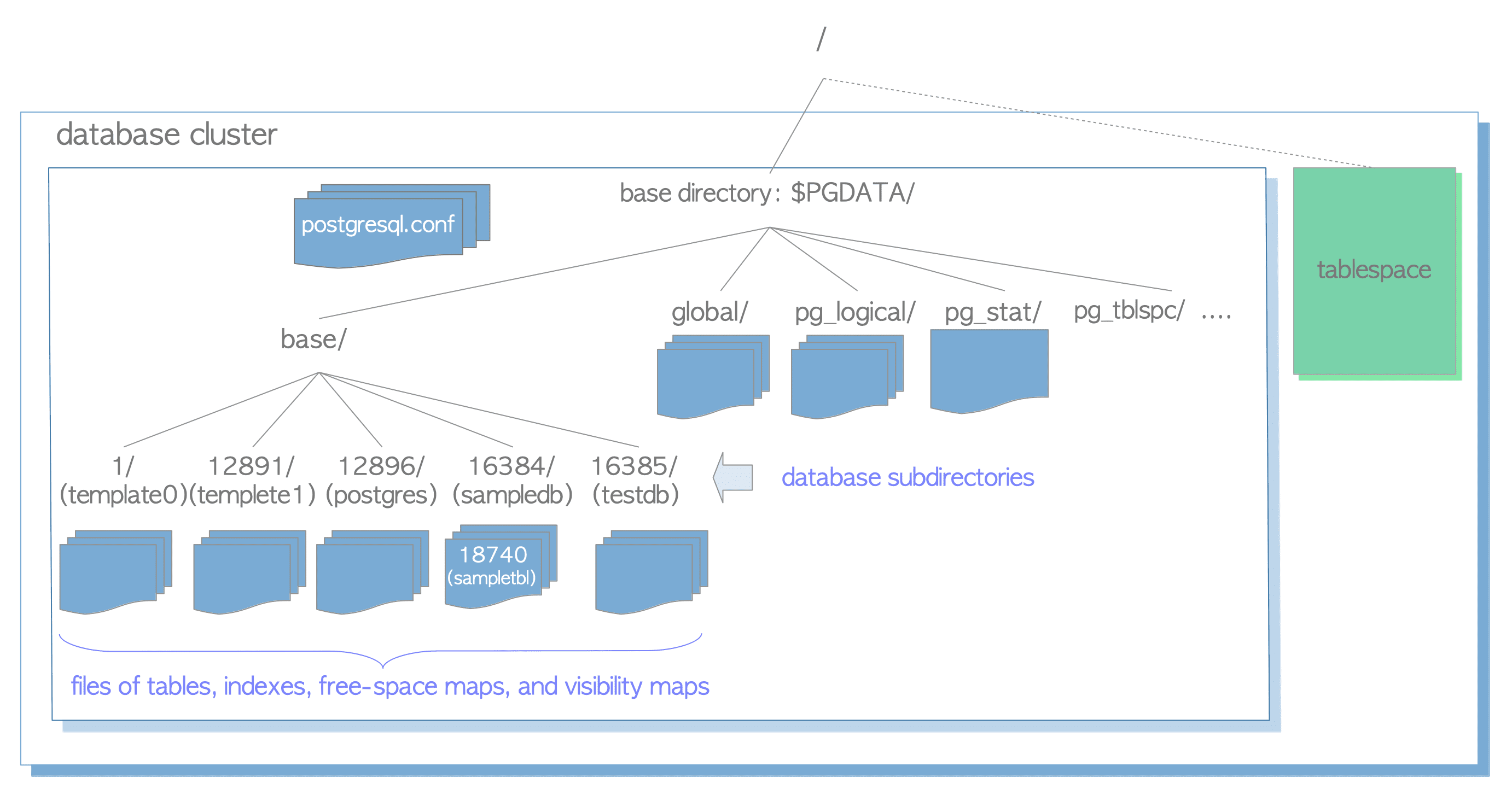


*Slika 2. prikaz identifikatora*

## 1.2. Fizička struktura

Klaster baza podataka u osnovi je jedan direktorijum koji se naziva osnovni direktorijum i sadrži neke poddirektorijume i veliki broj datoteka.

Na slici 3 dat je primer klastera baza podataka u PostgreSQL-u. Baza podataka je poddirektorijum ispod osnovnog poddirektorijuma, a svaka tabela i indeks predstavlja najmanje jednu datoteku koja se čuva u poddirektorijumu baze podataka kojoj pripada. Takođe postoji nekoliko poddirektorijuma koji sadrže određene podatke i konfiguracione datoteke. Iako PostgreSQL podržava *tablespaces*, značenje izraza se razlikuje od ostalih RDBMS-a. Tablespace u PostgreSQL-u je jedan direktorijum koji sadrži neke podatke izvan osnovnog direktorijuma.



*Slika 3. Primer klastera baze podataka*

## 1.2.2. Layout baza podataka

Baza podataka je poddirektorijum ispod osnovnog poddirektorijuma; a nazivi direktorijuma baze podataka su identični odgovarajućim OID-ovima. Na primer, kada je OID baze podataka elfakdb, ime njegovog poddirektorijuma je 202226.

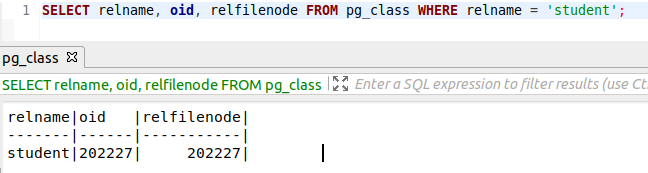
root@pc4:/var/lib/postgresql/11/main# ls -ld base/202226

drwx------ 2 postgres postgres 12288 апр 13 19:10 base/202226

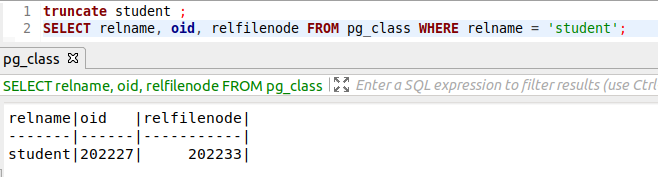
## Layout datoteka povezanih sa tabelama i indeksima

Svaka tabela ili indeks čija je veličina manja od 1GB je jedna datoteka koja se čuva u direktorijumu baze podataka kojoj pripada. Tabelama i indeksima kao objektima baze podataka interno se upravlja preko OID-a, dok tim datotekama podataka upravlja promenljiva, *relfilenode*. Vrednosti relfilenode tabela i indeksa u osnovi, ali ne uvek, odgovaraju njihovim identifikatorima.

Pokažimo OID i relfilenode tabele *student*:



Vrednosti relfilenode tabela i indeksa se menjaju izdavanjem nekih naredbi (npr. TRUNCATE, REINDEX, CLUSTER). Na primer, ako truncate-ujemo tabelu student, PostgreSQL će dodeliti tabeli novi relfilenode (202233).



Kada veličina tabela i indeksa pređe 1GB, PostgreSQL kreira novu datoteku koja se zove relfilenode.1 i koristi je. Ako je nova datoteka popunjena, kreiraće se sledeća nova datoteka pod nazivom relfilenode.2 itd.

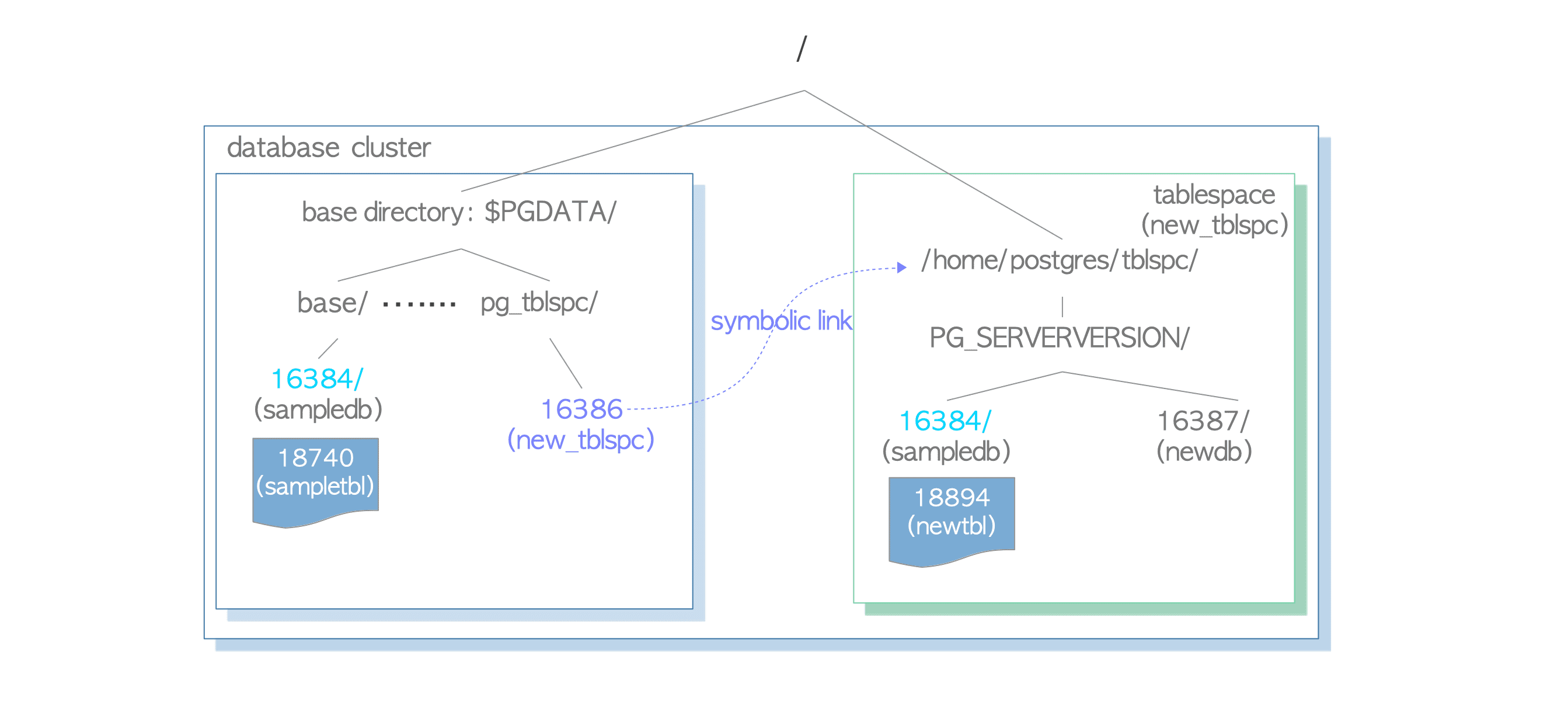
Maksimalna veličina datoteke tabela i indeksa može se promeniti korišćenjem konfigurace, opcijom  --with-segsize  prilikom porketanja PostgreSQL-a.

Svaka tabela ima dve povezane datoteke sa sufiksima '\_fsm' (free space map) i '\_vm' (visibility\_map). One sadrže podatke o veličini slobodnog prostora i vidljivosti na svakoj stranici u datoteci tabele. Indeksi imaju samo pojedinačne mape slobodnog prostora i nemaju mapu vidljivosti.

### 1.2.4. Tablespaces

Prostor tabela u PostgreSQL-u je dodatno područje podataka izvan osnovnog direktorijuma. Ova funkcija je implementirana u verziji 8.0.

Na slici 4 prikazan je unutrašnji raspored tablespace-ova i odnos sa glavnim područjem podataka.



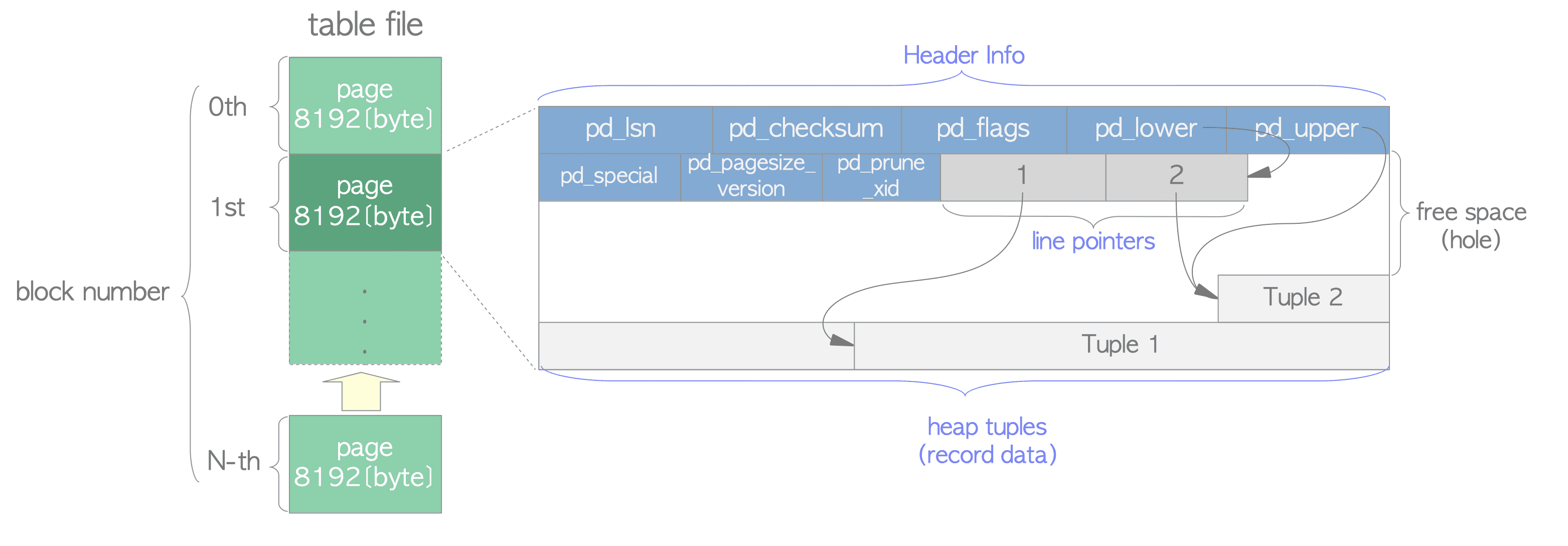
*Slika 4. Tablespace u klasteru baza podataka.*

Prostor tabela kreira se u direktorijumu kada se izvrši komanda CREATE TABLESPACE, a u njemu će se kreirati poddirektorijum specifičan za verziju (npr. PG\_14\_202104044). Način imenovanja za određenu verziju prikazana je u nastavku.

PG \_ 'Major version' \_ 'Catalogue version number'

## Interna struktura Heap Table fajla

Sadržaj datoteke sa podacima podeljena je na **stranice** (ili **blokove** ) fiksne dužine, a podrazumevana vrednost je 8192 bajta (8 KB). Te stranice unutar svake datoteke numerišu se od 0, a takvi se brojevi nazivaju **brojevima blokova**. Ako je datoteka popunjena, PostgreSQL dodaje novu praznu stranicu na kraj datoteke kako bi povećao veličinu datoteke.



*Slika 5. Izgled stranice datoteke heap tabele.*

Na slici 5 prikazan je izgled datoteke heap tabele.

Stranica na slici sadrži tri vrste podataka opisane na sljedeći način:

1. **Heap tuple** - predstavlja sam podatak o zapisu. Složeni su redom od dna stranice.
2. **Line pointers** - pokazivač na liniju veliki je 4 bajta i sadrži pokazivač na svaki heap tuple. Pokazivači linija čine jednostavan niz koji igra ulogu indeksa na heap tuple. Svaki indeks numeriše se sekvencijalno od 1 i naziva se offset brojem . Kada se na stranicu doda novi tuple, novi line pointer takođe se gura na niz da ukaže na novi tuple.
3. **Header data** – ovaj header se alocira na početku stranice. Veličina mu je 24 bajta i sadrži opšte informacije o stranici. Glavne varijable strukture opisane su u nastavku..
   * **pd\_checksum** - Ova varijabla čuva vrednost kontrolne sume ove stranice
   * **pd\_lower, pd\_upper** - pd\_lower pokazuje na kraj pokazivača na liniju, a pd\_upper na početak najnove hrpe hrpe.
   * **pd\_special** - Ova varijabla je za indekse. Na stranici unutar tabela pokazuje na kraj stranice. (Na stranici u indeksima pokazuje na početak posebnog prostora koji je područje podataka koje drže samo indeksi i sadrži određene podatke prema vrsti indeksa kao što su B-tree, GiST, GiN itd.

## 1.4 PostgreSQL system catalog

Sistemski katalozi su mesto gde relacioni sistem upravljanja bazom podataka čuva metapodatke šeme, kao što su informacije o tabelama i kolonama, i informacije o unutrašnjem radu.

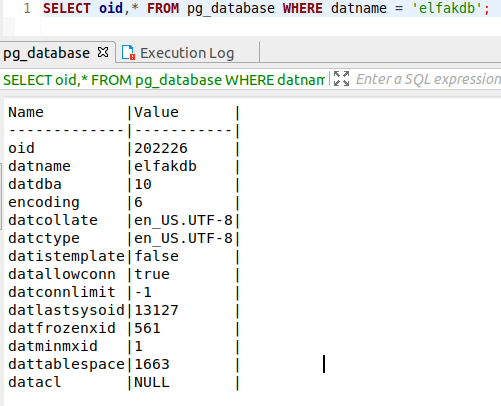
PostgreSQL sistemski katalozi su regularne tabele. Mogu se obrisati i ponovo kreirati, mogu im se dodati kolone, umetnuti i ažurirati vrednosti i na taj način ozbiljno poremettiti sistem. Obično sistemske kataloge ne treba menjati ručno, za to obično postoje SQL naredbe. (Na primer, CREATE DATABASE ubacuje red u katalog pg\_database - i zapravo kreira bazu podataka na disku.)

Postoje neki izuzeci za posebno ezoterične operacije, ali mnogi od njih su vremenom postali dostupni kao SQL naredbe.

Trenutno postoji preko 90 sistemskih kataloga od kojih su najkorisniji opisani u nastavku.

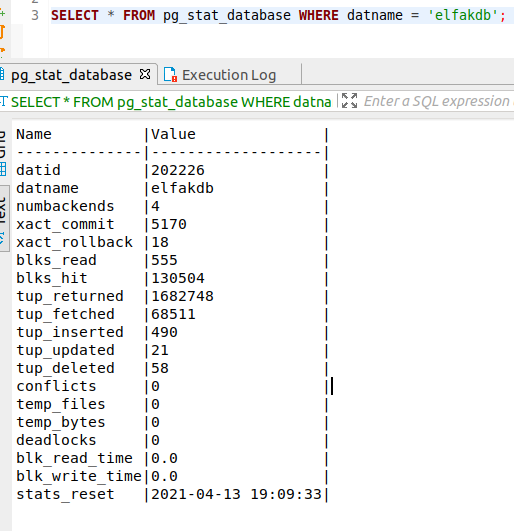
### Pg\_database

Tabela pg\_database sadrži red za svaku bazu podataka u klasteru, uključujući tri koje izlaze iz okvira (postgres, template0 i template1). Ovaj red sadrži informacije za kodiranje, ograničenje veze i druge osnovne metapodatke.



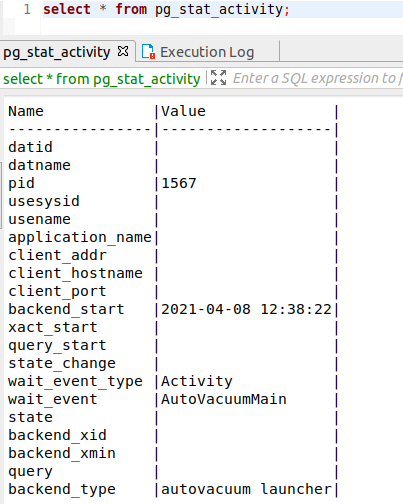
### Pg\_stat\_database

U ovoj tabeli sa statistikama dobijamo zanimljive i korisne podatke. Svaki red u ovoj tabeli sadrži aktivne podatke za svaku bazu podataka što može biti korisno u praćenju performansi baze podataka.



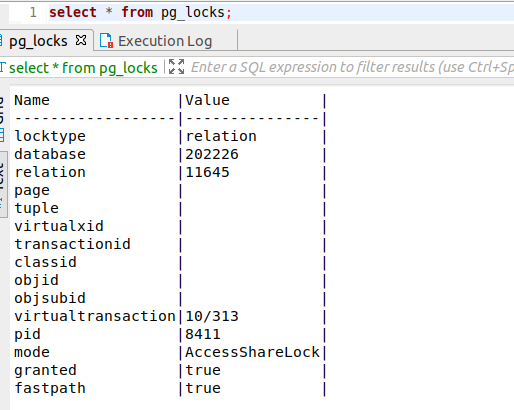
### Pg\_stat\_activity

pg\_stat\_activity je koristan view sistema PostgreSQL. Može se koristiti za analizu i rešavanje problema oko Postgres servera i oko izvršavanja kompleksnijih upita. Pogled pg\_stat\_activiti prikazuje red za svaki proces servera ili konekciju ka bazi.



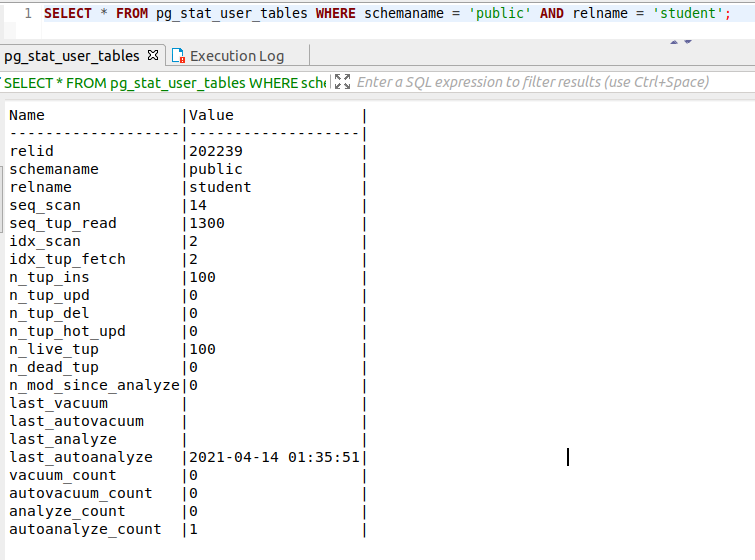
### Pg\_locks

Kad god se zaključa veza, te informacije se čuvaju u pg\_locks. Koristeći pid iz pg\_stat\_activity, možemo da upitamo pg\_locks da vidimo gde se nalaze lock-ovi i o kakvim lock-ovima je reč.



### Pg\_stat\_user\_tables

Ova tabela sadrži statistiku i informacije za svaku tabelu u određenoj bazi.



# **ORGANIZACIJA INDEKSA**

U PostgreSQL-u indeksi su posebni objekti baze podataka koji su uglavnom dizajnirani da ubrzaju pristup podacima. Oni su pomoćne strukture: svaki indeks se može izbrisati i ponovo stvoriti iz podataka u tabeli. Indeksi takođe služe za provođenje nekih ograničenja integriteta.

Trenutno je šest različitih vrsta indeksa ugrađeno u PostgreSQL. Uprkos svim razlikama između vrsta indeksa (koje se nazivaju i metode pristupa), svaki od njih na kraju pridružuje ključ (na primjer vrednost indeksiranog stupca) s redovima tablice koji sadrže ovaj ključ. Svaki red je identifikovan TID-om, koji se sastoji od broja bloka u datoteci i položaja reda unutar bloka. Ipak, s poznatim ključem ili nekim informacijama o njemu možemo brzo pročitati one slogove koji mogu sadržati informacije od interesa bez skeniranja cele tabele.

Važno je razumeti da indeks ubrzava pristup podacima uz određene troškove održavanja. Za svaku operaciju indeksiranih podataka, bilo da se radi o insertovanju, brisanju ili ažuriranju slogova tabele, indeksi za tu tabelu takođe se moraju ažurirati u istoj transakciji.

Proširivost povlači neke implikacije. Da bi se omogućilo jednostavno dodavanje nove metode pristupa sistemu, implementiran je interfejs mehanizma opšteg indeksiranja. Njegov glavni zadatak je dobiti TID-ove iz metode pristupa i raditi s njima.

## 2.1 Tipovi indeksa

PostgreSQL pruža mogućnost kreiranja nekoliko tipova indeksa:

* B-tree,
* Hash,
* GiST,
* SP-GiST,
* GIN i
* BRIN.

Svaki tip indeksa koristi drugačiji algoritam koji je najprikladniji za različite tipove upita. Podrazumevano, naredba CREATE INDEX kreira indekse B-stabla, koji odgovaraju najčešćim situacijama.

### 2.1.1 Btree (B-stabla)

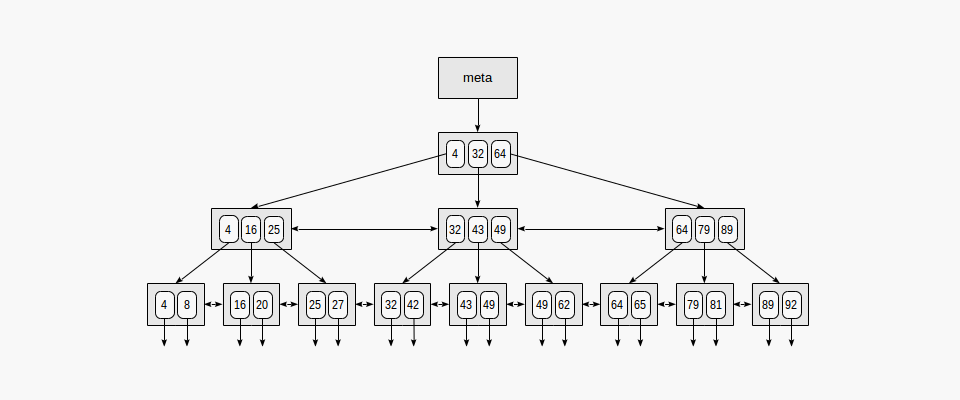
Tip indeksa B-stabla, pogodan je za podatke koji se mogu sortirati. Drugim rečima, za tip podataka moraju biti definisani operatori "veće", "veće ili jednako", "manje", "manje ili jednako" i "jednako". Treba imati na umu da se isti podaci ponekad mogu sortirati drugačije.

Redovi indeksa B-stabla spakovani su u stranice. Na stranicama lista, ovi redovi sadrže podatke za indeksiranje (ključevi) i reference na redove tablice (TID). Na internim stranicama svaki se red odnosi na podređenu stranicu indeksa i sadrži minimalnu vrednost na ovoj stranici.

B-stablo ima nekoliko važnih osobina:

* B-stabla su uravnotežena, odnosno svaka stranica lista odvojena je od korena istim brojem internih stranica. Stoga traženje bilo koje vrednosti traje isto vreme.
* B-stabla su višegranska, odnosno svaka stranica (obično 8 KB) sadrži puno (stotina) TID-ova. Kao rezultat toga, dubina B-stabala je prilično mala, zapravo do 4-5 za vrlo velike tabele.
* Podaci u indeksu razvrstavaju se prema opadajućem redosledu (i između stranica i unutar svake stranice), a stranice istog nivoa međusobno su povezane dvosmernim listom

Ispod, na slici 6, prikazan je pojednostavljeni primer indeksa nad jednom kolonom sa celobrojnim ključevima.



*Slika 6. primer indeksa nad jednom kolonom*

Prva stranica indeksa je metastranica koja se odnosi na koren indeksa. Interni čvorovi nalaze se ispod korena, a stranice listova u najnižem redu. Strelice predstavljaju reference sa čvorova listova na slogove tabela (TID).

Primer kreiranja indeksa



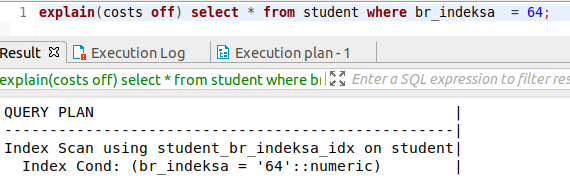
#### **Pretraga po jednakosti**

Razmotrimo pretragu vrednosti u stablu prema uslovu "indexed-field = expression ". Recimo, zanima nas ključ 49.

Pretraga započinje s korenskim čvorom i potrebno je odrediti na koji se od podređenih čvorova spustiti. Budući da se znaju ključevi u korenskom čvoru (4, 32, 64), stoga otkrivamo raspone vrednosti u podređenim čvorovima. Budući da je 32 ≤ 49 <64, moramo se spustiti do drugog podređenog čvora. Dalje, isti postupak se rekurzivno ponavlja sve dok ne dođemo do čvorišta odakle se mogu dobiti potrebni TID.

U stvarnosti, brojne pojedinosti komplikuju ovaj naizgled jednostavan postupak. Na primer, indeks može sadržati nejedinstvene ključeve i može biti toliko jednakih vrednosti da ne odgovaraju jednoj stranici. Vraćajući se na naš primer, čini se da bismo se trebali spustiti s unutarnjeg čvora preko reference na vrednost 49. Ali, kao što je jasno sa slike, na ovaj ćemo način preskočiti jednu od tipki "49" na prethodnoj stranici lista .

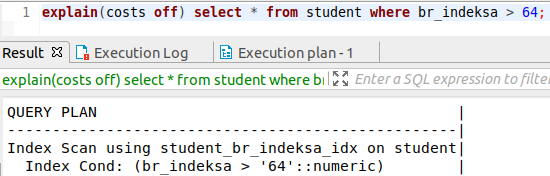
Druga je komplikacija što tokom pretraživanja drugi procesi mogu promeniti podatke, stablo se može obnoviti, stranice se mogu podeliti na dve itd. Svi algoritmi su dizajnirani da ove istovremene operacije ne ometaju jedna drugu i ne uzrokuju dodatne blokade gde god je to moguće.



#### **Pretraga po nejednakosti**

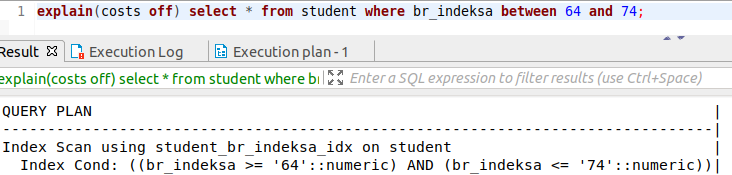
Kada pretražujemo prema uslovu " indeksirano polje ≤ izraz " (ili " indeksirano polje ≥ izraz "), prvo pronalazimo vrednost (ako postoji) u indeksu prema uslovu jednakosti " indeksirano polje = izraz ", a zatim prolazimo kroz stranice u odgovarajućem smeru do kraja.

Operatori "veći" i "manji" podržani su na sličan način, osim što prvobitno pronađena vrednost mora ispustiti.



#### **Pretraga po opsegu**

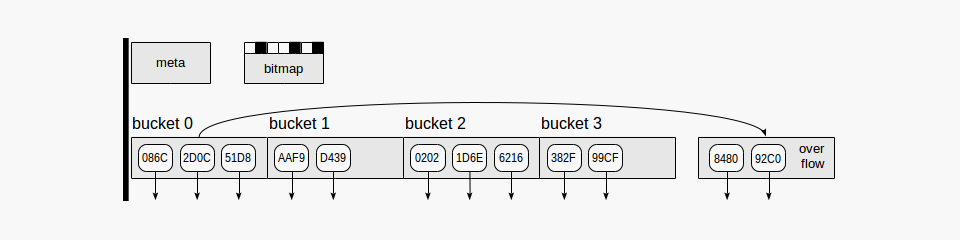
Kada pretražujemo po opsegu " izraz1 ≤ indeksirano polje ≤ izraz2 ", nalazimo vrednost prema stanju " indeksirano polje = izraz1 ", a zatim nastavljamo šetati stranicama lista dok je zadovoljen uslov " indeksirano polje ≤ izraz2 ", ili obrnuto: počinje se sa drugim izrazom i kreće se u suprotnom smeru dok se ne dođe do prvog izraza.



### 2.1.2 Hash

Kao što im ime kaže, ovi indeksi koriste hash funkciju za mapiranje podataka. Ta funkcija mapira sve vrednosti u bazi u 32-bitni integer, tj. u hash code.

Sve informacije i svi indeksi moraju biti organizovani u stranice. Takve indeksne stranice se smeštaju u cache buffer i pozivaju se odatle na isti način kao i stranice tabela.



*Slika 7. Struktura hash indeksa*

Hash indeks, kao što je prikazano na slici 7, koristi četiri vrste stranica (sivi pravougaonici), a to su:

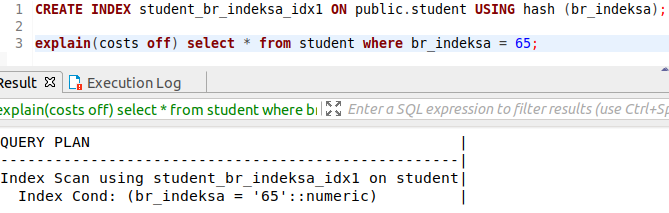
* **meta** stranica – sadrži informacije o tome šta se nalazi u indeksu
* **bucket** stranica – glavna stranica indeksa, sadrži parove *hash code – TID*
* **overflow** stranica – ima istu strukturu kao bucket i koristi se kada je jedna stranica nedovoljna za ceo bucket
* **bitmap** stranica - vodi računa o overflow stranicama koje su trenutno prazne i mogu biti korišćene za druge bucket-e.

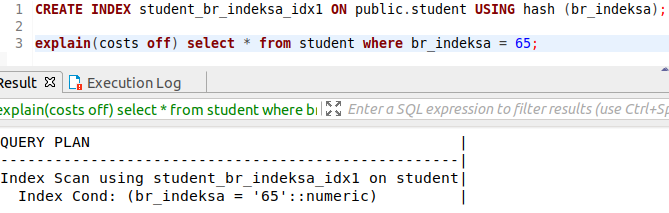
Strelice prikazuju referenece na slogove tablice.

Treba imati na umu da se ne može smanjiti veličina hash indeksa. Ako izbrišemo neke indeksirane slogove, stranice koje su jednom dodijeljene neće se vratiti u operativni sistem, već će se ponovo koristiti za nove podatke nakon vakumiranja. Jedina opcija za smanjenje veličine indeksa je ponovno kreiranje pomoću naredbe REINDEX ili VACUUM FULL.

Hash indeksi generalno mogu pružiti brže pretraživanje i vreme kreiranje od B-Tree indeksa. Veliki problem kod njih je što su ograničeni samo na operatore jednakosti, tako da morate tražiti tačna podudaranja. To čini hash indekse daleko manje fleksibilnima od najčešće korišćenih B-Tree indeksa.

Primer:





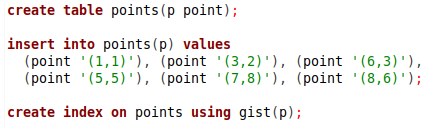
### 2.1.3 GiST

GiST je skraćenica od "generalizovanog stabla pretraživanja". Ovo je uravnoteženo stablo pretraživanja, baš kao i "btree".

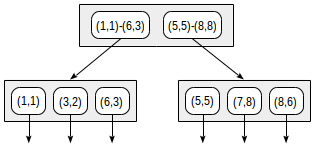
Koja je razlika? Indeks "btree" strogo je povezan sa semantikom poređenja: podrška za "veće", "manje" i "jednake" operatore. Međutim, moderne baze podataka čuvaju tipove podataka za koje ovi operatori jednostavno nemaju smisla: geopodaci, tekstualni dokumenti, slike, ...

GiST je stablo uravnoteženo po visini koje se sastoji od čvorova koji predstavljaju stranice. Čvorovi se sastoje od indeksnih redove.

Svaki red u čvoru sadrži neki predikat (logički izraz) i referencu na slog tabele (TID). Indeksirani podaci (ključ) moraju odgovarati ovom prediktu.

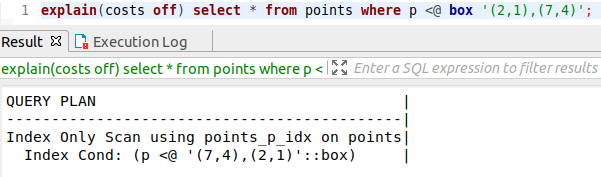


Struktura indeksa iz primera iznad izgleda kao na slici 8:

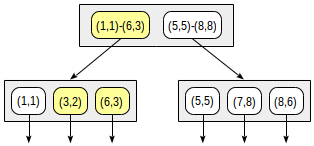


*Slika 8.*

Ovako kreirani indeks može se koristiti za ubrzavanje sledećeg upita, na primer: "naći sve tačke koje se nalaze u zadatom pravougaoniku". Ovakav uslov se može prikazatzi kao: p <@ box '(2,1),(6,3)'(operator <@ predstavlja "sadržan u"):



Pretraga počinje od korena. Pravougaonik (2,1) - (7,4) seče se sa (1,1) - (6,3), ali se ne seče sa (5,5) - (8,8), stoga, nema potrebe da se spusti do drugog podstabla.



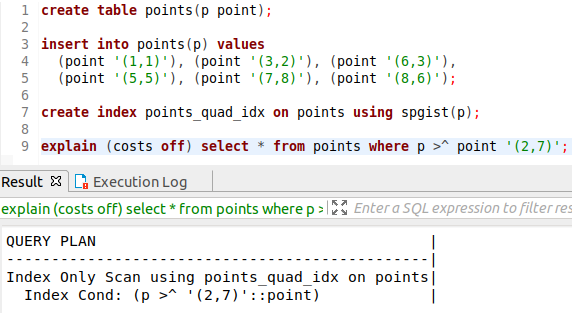
*Slika 9. Pretraga pomoći GiST*

Došavši do čvora lista, prolazimo kroz tri tačke koje se tamo nalaze i kao rezultat vraćamo dve od njih: (3,2) i (6,3).

2.1.4 SP-GiST (Space Partitioning)

SP-GiST je skraćenica za prostorno podeljen GiST. SP-GiST koristi particionisana stabla pretraživanja, koja olakšavaju razvoj širokog spektra različitih nebalansiranih struktura podataka, poput četvorostrukih stabala, k-d stabala i radix stabala. Zajednička karakteristika ovih struktura je da oni neprestano dele prostor za pretragu na particije koje ne moraju biti jednake veličine. Pretrage koje se dobro podudaraju sa pravilom podele mogu biti vrlo brze.

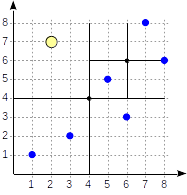
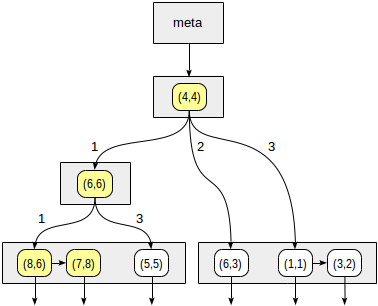
Ove popularne strukture podataka prvobitno su razvijene za upotrebu u memoriji. U glavnoj memoriji su obično dizajnirane kao skup dinamički dodeljenih čvorova povezanih pointerima. Ovo nije pogodno za direktno skladištenje na disku, jer ovi lanci pointera mogu biti prilično dugi što bi zahtevalo previše pristupa disku. Suprotno tome, strukture podataka zasnovane na disku trebalo bi da imaju veliku količinu ventilatora da bi smanjile I / O.



Na primer, ako treba pronaći sve tačke koje leže iznad date tačke, može se izvršiti upit s uslovom p> ^ point '(2,7)' .

Počinjemo sa korenskim čvorom i koristimo funkciju poređenja da biramo na koje će se podređene čvorove spustiti. Za operator >^, ova funkcija upoređuje tačku (2,7) sa centralnom tačkom čvora (4,4) i bira kvadrante koji mogu sadržati tražene tačke, u ovom slučaju prvi i četvrti kvadrant. (Slika 10)

U čvoru koji odgovara prvom kvadrantu ponovo određujemo podređene čvorove pomoću funkcije poređenja. Centralna tačka je (6,6) i opet treba da pogledamo prvi i četvrti kvadrant.

Slika 10. Slika 11.

Lista čvorova lista (8,6) i (7,8) odgovara prvom kvadrantu, od kojih samo tačka (7,8) ispunjava uslov upita. Referenca na četvrti kvadrant je prazna. (Slika 11)

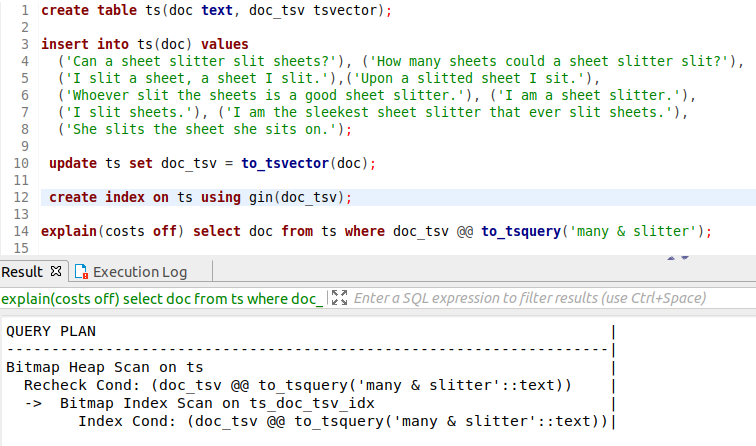
U unutrašnjem čvoru (4,4) referenca na četvrti kvadrant je takođe prazna, što završava pretragu.

2.1.5 GIN

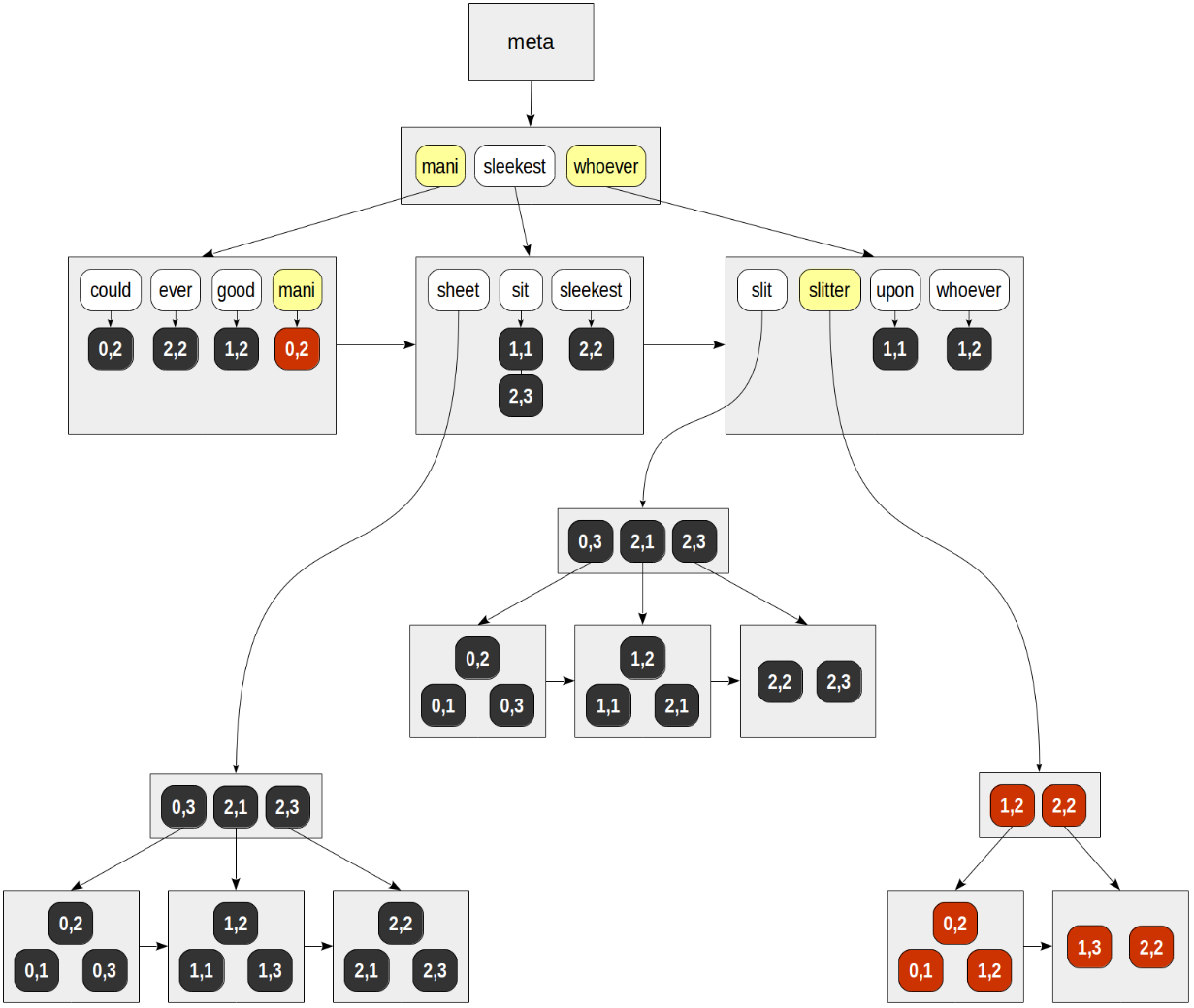
GIN je skraćenica za Generalizovani invertovani indeks. Ovo je takozvani obrnuti indeks. On manipuliše tipovima podataka čije vrednosti nisu atomične, već se sastoje od elemenata. Ove vrste ćemo nazvati složenicama. I to nisu vrednosti koje se indeksiraju, već pojedinačni elementi; svaki element upućuje na vrednosti u kojima se javlja.

Dobra analogija sa ovom metodom je indeks na kraju knjige koji za svaki pojam daje spisak stranica na kojima se taj pojam javlja. Način pristupa mora osigurati brzo pretraživanje indeksiranih elemenata, baš kao i indeks u knjizi. Uređeni skup referenci na redove tabele koji sadrže složene vrednosti sa elementom povezan je sa svakim elementom. Uređenost nije bitna za pronalaženje podataka (redosled sortiranja TID-a ne znači mnogo), ali je važan za unutrašnju strukturu indeksa.

Glavno područje primene GIN metode je ubrzanje pretraživanja celog teksta, što pokazuje i sledeći primer.

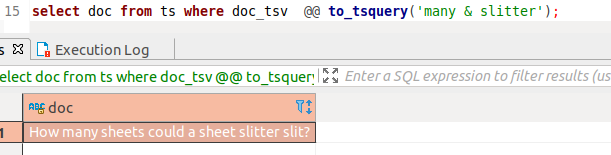


Struktura ovakve pretrage po indeksu predstavljena je na slici 12.



Slika 12.

Konačno, za svaki pronađeni TID poziva se funkcija doslednosti API-ja koja mora odrediti koji se od pronađenih redova podudara sa upitom za pretragu. Budući da se leksemi u našem upitu pridružuju logičkim «i», jedini vraćeni red je (0,2), tj:

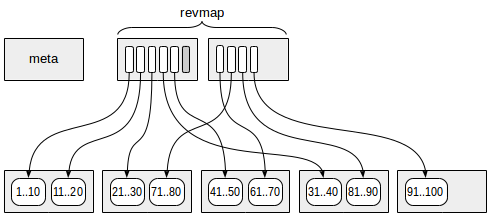


2.1.6 BRIN

Za razliku od indeksa sa kojima smo se već upoznali, ideja BRIN-a je da izbegne pregledavanje definitivno neprikladnih redova umesto da brzo pronađe odgovarajuće. Ovo je uvek netačan indeks: on uopšte ne sadrži TID-ove redova tabele.

Pojednostavljeno, BRIN dobro funkcioniše za kolone gde vrednosti odgovaraju njihovoj fizičkoj lokaciji u tabeli. Drugim rečima, ako upit bez ORDER BY vraća sortirane vrednosti.

Ovo radi na sledeći način. Tabela je podeljena na opsege od nekoliko stranica (ili nekoliko blokova) - otuda i naziv: Block Range Index, BRIN. Indeks čuva sažete informacije o podacima u svakom opsegu. To su po pravilu minimalne i maksimalne vrednosti, ali se dešava da budu različite, kao što je prikazano. Pretpostavimo da se izvodi upit koji sadrži uslov za kolonu; ako tražene vrednosti ne uđu u interval, može se preskočiti čitav opseg; ali ako to dobiju, moraće se pregledati svi redove u svim blokovima da bi se među njima izabrao ona koji se podudara.



Struktura BRIN-a

Prva (tačnije, nula) stranica sadrži metapodatke.

Stranice sa sažetim informacijama nalaze se na određenom odmaku od metapodataka. Svaki red indeksa na tim stranicama sadrži sažete informacije o jednom opsegu.

Između meta stranice i sažetih podataka nalaze se stranice sa mapom obrnutog opsega (skraćeno kao «revmap»). Zapravo, ovo je niz pokazivača (TID) na odgovarajuće indeksne redove.

Za neke opsege, pokazivač u «revmap» može dovesti do nijednog indeksnog reda (jedan je na slici označen sivom bojom). U takvom slučaju, smatra se da opseg još nema sažete informacije.

## 2.2 Parcijalni indeksi

Parcijalni indeks pokriva samo podskup podataka tabele. To je indeks sa WHERE uslovom. Ideja je povećati efikasnost indeksa smanjenjem njegove veličine. Manji indeks zauzima manje prostora za skladištenje, lakši je za održavanje i brži sa skeniranje.

Na primeru pomenute STUDENT tabele mozemo kreirati indeks nad studentima sa manjim brojem indeksa od 20:



Ovakav indeks ostaće mali po veličini, a može se koristiti u nekim kompleksnijim upitima.

## 2.2 Expression indeksi

Expression indeksi korisni su za upite koji se podudaraju sa nekom funkcijom ili modifikacijom podataka. Postgres omogućava da indeksiranje rezultata te funkcije tako da pretrage postanu efikasne poput pretraživanja prema vrednostima neobrađenih podataka. Na primer, moguće je pretraživanje studenata po imenu zanemarujući velika i mala slova. U tom slučaju je moguće sačuvati ime studenta takvim kakvo jeste, ali se pretražuje WHERE lower(ime) = '<lowercased-ime>'.. Jedini način da se indeks koristi u takvom upitu je pomoću expression indeksa izraza poput ovog:



## 2.2 Unique indeksi

Jedinstveni indeks garantuje da tabela neće imati više od jednog sloga iste vrednosti. Povoljno je kreirati jedinstvene indekse iz dva razloga: integritet podataka i performanse. Pretrage po jedinstvenom indeksu su uglavnom vrlo brze.

U pogledu integriteta podataka, upotreba validates\_uniqueness\_of validacije na klasi ActiveModel zapravo ne garantuje jedinstvenost, jer mogu i biće istovremeni korisnici koji kreiraju nevažeće zapise. Zbog toga bi uvek trebalo kreirati constraint na nivou baze podataka - bilo indeksom ili jedinstvenim constraint-om.

Mala je razlika između jedinstvenih indeksa i jedinstvenih constraint-a. Jedinstveni indeksi mogu se smatrati nižim nivoom, jer expression i parcijalni indeksi ne mogu se kreirati kao jedinstvena ograničenja. Mogući su čak i parcijalni jedinstveni ekspresioni indeksi.

# **LITERATURA**

1. Indexes in POstgreSQL; Egor Rogov; Dostupno na: <https://postgrespro.com/blog/pgsql/3994098>
2. Efficient Use of PostgreSQL Indexes; Dostupno na: <https://devcenter.heroku.com/articles/postgresql-indexes>
3. The Internals of PostgreSQL; Dostupno na: <https://www.interdb.jp/pg/pgsql01.html>
4. The Internals of PostgreSQL; Dostupno na: <https://www.interdb.jp/pg/pgsql01.html>
5. PostgreSQL documentation; Dostupno na: <https://www.postgresql.org/docs/13/internals.html>