Elektronski fakultet u Nišu

SEMINARSKI RAD

SISTEMI ZA UPRAVLJANJE BAZAMA PODATAKA

INTERNA ORGANIZACIJA I STRUKTURA SKLADIŠTA PODATAKA KOD MYSQL BAZE PODATAKA

Student:  
Vukadin Drašković, br. indeksa 1613

Sadržaj

[1. Uvod 3](#_Toc163997739)

[2. Sistem za upravljanje bazom podataka 4](#_Toc163997740)

[2.1. Arhitektura sistema za upravljanje bazom podataka 4](#_Toc163997741)

[2.2. Pregled nekih Storage Engine-a koje podržava MySQL DBMS 6](#_Toc163997742)

[2.2.1. InnoDB 6](#_Toc163997743)

[2.2.2. MyISAM 7](#_Toc163997744)

[2.2.3. Memory 7](#_Toc163997745)

[2.2.4. Blackhole 8](#_Toc163997746)

[3. InnoDB Storage Engine 9](#_Toc163997747)

[3.1. In-Memory komponente 10](#_Toc163997748)

[3.1.1. Buffer Pool 10](#_Toc163997749)

[3.1.2. Change Buffer 11](#_Toc163997750)

[3.1.3. Adaptive Hash Index 13](#_Toc163997751)

[3.1.4. Log Buffer 13](#_Toc163997752)

[3.2. On-Disk komponente 14](#_Toc163997753)

[3.2.1. System Tablespace 16](#_Toc163997754)

[3.2.2. File-Per-Table Tablespace 17](#_Toc163997755)

[3.2.3. General Tablespace 18](#_Toc163997756)

[3.2.4. Undo Tablespace 20](#_Toc163997757)

[3.2.5. Temporary Tablespace 20](#_Toc163997758)

[3.2.6. Doublewrite Buffer 21](#_Toc163997759)

[3.2.7. Redo Log 23](#_Toc163997760)

[4. Zaključak 25](#_Toc163997761)

[5. Literatura 26](#_Toc163997762)

# Uvod

Baze podataka predstavljaju glavnu komponentu svakog softverskog sistema, jer su one zadužene za čuvanje samih podataka koje koristi aplikacija. Iako baza podataka ima veliki značaj za sistem, način na koji ona organizuje i čuva podatke je nepoznat mnogim inženjerima, zato što im se pristupa preko unapred određenih interfejsa koji su smišljeni tako da maksimalno apstrahuju internu organizaciju i strukturu same baze.

Cilj ovog rada je upoznavanje sa komponentama koje baza podataka koristi za čuvanje podataka, načinom na koji ih čuva kao i organizacijom samih podataka na primeru MySQL sistema za upravljanje bazom podataka. Prvo će biti ukratko opisani osnovni pojmovi vezani za baze podataka, nakon čega će se opisati arhitektura MySQL-a, a zatim će se opisati komponente koje učestvuju u internoj strukturi i organizaciji skladišta podataka MySQL-a.

# Sistem za upravljanje bazom podataka

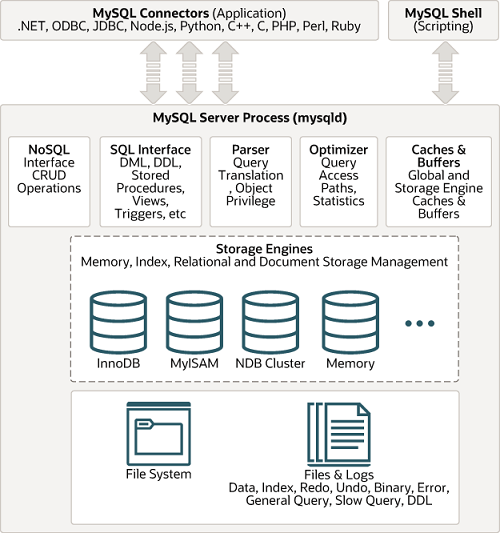
Baza podataka (engl. *database*) predstavlja skup logički grupisanih podataka ili informacija koje se skladište u memoriji ili na disku. Treba napomenuti da baza podataka sama po sebi predstavlja isključivo podatke, a ne i softver koji nam omogućava da izvršavamo određene operacije nad tim podacima – da ih preuzimamo pomoću SQL (engl. *Structured Query Language*) upita, ili da ih ažuriamo DML (engl. *Data Manipulation Language*) naredbama. Softver koji nam omogućava da komuniciramo sa bazom podatka, i da upravljamo istom, se zove sistem za upravljanjem bazom podataka (eng. *Database Management System* - *DBMS*). DBMS predstavlja interfejs između baze podataka i korisnika baze podataka, što se može videti na slici 1.

  
Slika 1. DBMS predstavlja interfejs baze podataka [1]

Sistemi za upravljanje bazama podataka se u osnovi mogu podeliti na tip baze za koju su namenjeni. Tako imamo relacione sisteme za upravljanje bazom podataka (engl. *Relational Database Management System - RDBMS)* koji se koriste za upravljanje relacionim bazama podataka, i NoSQL (engl. *Not Only SQL*) DMBS-ove koji se koriste za odgovarajuće NoSQL baze podataka. Najpopularniji relacioni DBMS-ovi su MySQL, SQL Server, Oracle Database, PostgreSQL... Od NoSQL DBMS-ova, najviše se izdvajaju Redis, MongoDB, Neo4j i Cassandra.

## Arhitektura sistema za upravljanje bazom podataka

DBMS, kao i svaki drugi softver, ima svoju arhitekturu koja se sastoji od određenih komponenti. Ovaj rad će biti baziran na MySQL relacionom sistemu za upravljanje bazom podataka, tako da će on biti uzet kao primer arhitekture nekog DBMS-a, koja je prikazana na slici 2 [2]. Može se uočiti da su klijentski i serverski deo sistema jasno razdvojeni, pri čemu arhitektura serverskog procesa *mysqld* predstavlja upravo arhitekturu MySQL DBMS-a. DBMS nudi interfejs korisnicima za preuzimanje i manipulaciju podataka, gde se nakon izdavanja komande od strane klijenta ona prevodi u format koji je pogodan za dalju obradu, što je posao *Parser* komponente arhitekture. Veliku ulogu u svakom DBMS-u igra *Optimizer* komponenta jer je ona zadužena da se na optimalan način izvrši komanda koju je izdao korisnik. Tu se nalaze i razni keševi i baferi koje koristi DBMS da bi keširao rezultate upita, kako se ne bi kontaktiralo skladište podataka svaki put ukoliko je rezultat izvršenja istog upita u međuvremenu ostao nepromenjen.

  
Slika 2. Arhitektura MySQL DBMS-a [2]

Komponenta koja je zadužena za organizaciju podataka, manipulaciju njima i čuvanje istih na fajl sistemu, odnosno na disku, je upravo *Storage Engine* komponenta ove arhitekture. Zbog toga se za Storage Engine može reći da predstavlja srž svakog DBMS-a, jer je odgovoran za upravljanje samim podacima. Kako je tema seminarskog rada interna struktura i organizacija skladišta podataka kod MySQL baze podataka, upravo je Storage Engine komponenta kojom ćemo se baviti u ovom radu. MySQL je DBMS koji nudi korisniku mogućnost korišćenja različitih Storage Engine-a. Svaki od njih ima svoje prednosti i mane.

## Pregled nekih Storage Engine-a koje podržava MySQL DBMS

Storage Engine-i koje podržava MySQL [3] se mogu videti ukoliko se u MySQL terminalu izvrši komanda prikazana na slici 3.

Slika 3. Izvršenje komande *SHOW ENGINES;* koja pokazuje tabelu Storage Engine-a koje podržava MySQL

### InnoDB

*InnoDB* je Storage Engine koji korisniku nudi podršku za ACID transakcije. Transakcija predstavlja skup akcija, odnosno naredbi, koji se treba izvršiti nad bazom podataka. ACID predstavlja akronim koji se odnosi na četiri svojstva koja treba da ispuni transakcija, a to su:

* **Atomičnost** (engl. *Atomicity*) – odnosi se na to da će se ili izvršiti sve naredbe koje čine transakciju, ili se nijedna neće izvršiti,
* **Konzistentnost** (engl. *Consistency*) – sistem nakon izvršenja transakcije prelazi u novo konzistentno, odnosno validno stanje,
* **Izolacija** (engl. *Isolation*) – sve akcije koje se izvršavaju unutar transkacije nisu vidljive drugim transakcijama, sve dok se transakcija ne finalizuje, i
* **Trajnost** (engl. *Durability*) – kada se jednom izvrši transakcija, novo stanje je trajno i podaci se ne mogu vratiti u stanje koje su imali pre izvršenja transakcije.

Pored podrške za transakcije, InnoDB omogućava oporavak podataka prilikom iznenadnog pada baze podataka (ima *crash-recovery* svojstvo).

Kada transakcija pristupa podacima unutar relacije, neophodno je da ti podaci budu zaključani (engl. *locked*) za tu transackiju, kako se njhovo stanje ne bi menjalo dok se transakcija ne završi. Ovim zaključavanjem se podaci izoluju od ostalih transakcija. Kod InnoDB-a, granularnost izolacije podataka je na nivou slogova kojima se pristupa, tako da ostale transakcije mogu da konkurentno pristupaju ostalim podacima u istoj relaciji. Ovaj tip izolacije (odnosno „zaključavanja“ podataka) se naziva *row-level locking* (zaključavanje na nivou vrste).

InnoDB smešta podatke korisnika u klasterovanom indeksu (engl. *clustered* *index*) koji je indeksiran primarnim ključem. Dok je indeks struktura baze podataka koja omogućava da se na efikasan način pristupa podacima preko indeksiranih kolona unutar jedne relacije, klasterovani indeks predstavlja indeks kod koga je redosled podataka na koji on pokazuje jednak redosledu podataka na samom disku na kome su podaci smešteni. U suprotnom je indeks neklasterovan (engl. *unclustered* *index*).

Pored svega navedenog, kako bi obezbedio integritet podataka u bazi, InnoDB podržava ograničenja vezana za FOREIGN KEY (srp. strani ključ). Prilikom manipulacije nad podacima koji imaju FOREIGN KEY ograničenje se proverava da li FOREIGN KEY ima validnu vrednost primarnog ključa iz relacije koju referencira.

### MyISAM

Tabele *MyISAM* [4] Storage Engine-a zauzimaju veoma malo prostora na disku, što je dobra stvar. Loša stvar je u tome što za razliku od InnoDB-a koji ima *row-level locking* mehanizam, MyISAM izoluje podatke na nivou cele tabele (engl. *table-level locking*), tako da ukoliko jedna transakcija pristupi samo jednom redu relacije, ostale transakacije neće moći da pristupe podacima sve dok se transakcija ne završi. Zato je MyISAM zgodan da se koristi u aplikacijama kod kojih je broj naredbi za ažuriranje zanemarljiv u odnosu na broj naredbi koje treba čitaju podatke iz baze.

### Memory

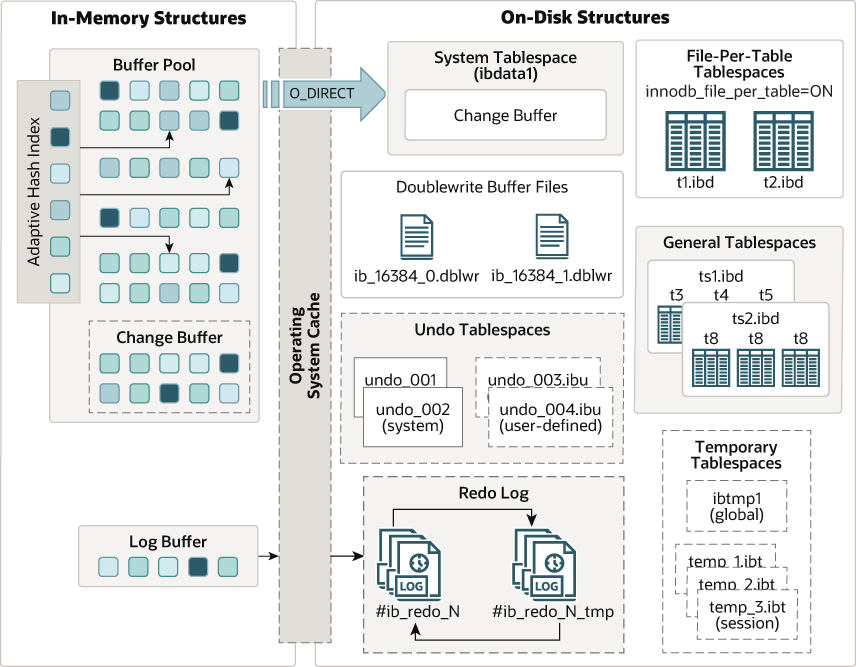
*Memory* Storage Engine sve podatke smešta u RAM memoriju računara na kome se DBMS izvršava, tako da omogućava veoma brz pristup podacima. Ovaj Engine se koristi isključivo kada podacima treba da se pristupa samo za potrebe operacije čitanja.

### Blackhole

*Blackhole* (srp. *crna rupa*) je Storage Engine koji ne skladišti podatke nigde[[1]](#footnote-1) – prihvata ih prilikom upisa, ali ih ne perzistira na fajl sistemu. Najčešće se koristi u distribuiranom okruženju, na primer na serveru koji za cilj ima isključivo čuvanje logova, odnosno komanda koje želi da izvrši korisnik, kako bi komande bile validno prosleđene serveru koji ima podatke.

# InnoDB Storage Engine

InnoDB je podrazumevani Storage Engine za MySQL u najnovijoj verziji 8.3. Arhitektura InnoDB-a [5] je prikazana na slici 4. Uočavamo da se u gruboj podeli komponente InnoDB-a mogu podeliti na komponente koje se nalaze u radnoj memoriji tokom izvršenja (engl. *In-Memory Structures*), i komponente koje se nalaze na disku (engl. *On-Disk Structures*). Iz prethodno navedenog se može reći da se In-Memory komponente koriste za optimizaciju rada Storage Engine-a, jer je pristup podacima iz memorije brži nego pristupu sa diska, dok se On-Disk komponente koriste za perzistenciju podataka – za potrebe korisnika i metapodataka – za potrebe pravilnog funkcionisanja InnoDB-a.

Slika 4. Arhitektura InnoDB Storage Engine-a

## In-Memory komponente

Od In-Memory komponenti sa slike 4 se mogu izdvojiti *Buffer* *Pool*, *Change* *Buffer*, *Adaptive* *Hash* *Index* i *Log* *Buffer*.

### Buffer Pool

Buffer Pool [5] predstavlja oblast u radnoj memoriji gde InnoDB kešira odgovarajuće stranice tabela i indeksa kojima je korisnik pristupio, čime se omogućava da se ponovnim pristupanjem istih podataka u skorije vreme oni pribave iz Buffer Pool-a, odnosno radne memorije. Zanimljivo je da kapacitet radne memorije koju koristi Buffer Pool, na serverima koji isključivo nude usluge baze podataka, može ići do 80%.

Buffer Pool je u osnovi lančana lista čiji su elementi stranice koje sadrže podatke sa diska. Lančana lista je semantički podeljena u dva dela:

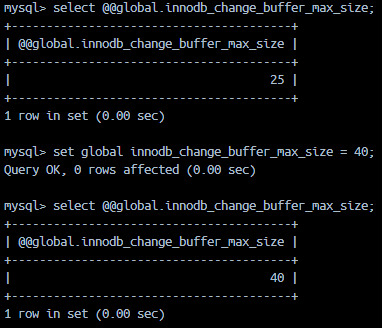
* podlista novih stranica (nova podlista) – nalazi se na početku lančane liste i označava stranice kojima se skoro pristupilo,
* podlista starih stranica (stara podlista) – nalazi se na kraju i sadrži stranice kojima se nije pristupilo neko vreme.

Buffer Pool funkcioniše po LRU (engl. *Last Recently Used*) principu, gde se za potrebe dodavanja nove stranice u listu, ukoliko je lista puna, iz iste izbacuje stranica kojoj se najduže nije pristupalo. Ova stranica je poslednja u podlisti starih stranica. Nova stranica se dodaje između dve podliste, tako da nakon dodavanja pripada starim stranicama. Da bi postala deo nove podliste, neophodno je da korisnik pristupi nekim podacima koje ona sadrži, što znači da je moguće da se u Buffer Pool učita stranica koja sadrži podatke koje korisnik nije zahtevao. Postavlja se pitanje: zašto bi stranica bila učitana u Buffer Pool ukoliko korisnik nije zahtevao pristup podacima koje ona sadrži? Odgovor leži u *read-ahead* (srp. čitanje unapred) principu koji koristi InnoDB prilikom učitavanja stranica. On funkcioniše tako što InnoDB na osnovu prethodnih čitanja, može da pretpostavi koje će naredne podatke korisnik zahtevati i da unapred pribavi stranice koje sadrže te podatke. Ukoliko je stranica učitana u Buffer Pool kao posledica čitanja unapred, može se desiti da joj korisnik nikad ne pristupi i zato se sve učitane stranice inicijalna stavljaju u podlistu starih stranica. Pak, ukoliko je stranica učitana jer je korisnik zahtevao neke podatke koje ona sadrži, biće smeštena u podlistu starih stranica, a neposredno nakon čitanja njenih podataka će biti prebačena u novu podlistu. Pristupanje stranici koja se nalazi u Buffer Pool-u, bilo da se ona nalazi u starom ili novom delu liste, rezultuje pomeranjem stranice na početak nove podliste.

### Change Buffer

Change Buffer [6] predstavlja In-Memory komponentu koja ima za cilj da kešira promene nad podacima koji se ne nalaze u Buffer Pool-u. Razlog tome je što nema potrebe da se podaci preuzimaju sa diska, zatim promene i upišu nazad na disk ukoliko se ne zahteva i njihovo čitanje, jer bi to bila previše skupa operacija. Promene će se nad odgovarajućim slogovima u bazi izvršiti tek kada budu pročitani, odnosno učitani u Buffer Pool. Naredbe koje se keširaju su DML naredbe INSERT, UPDATE i DELETE.

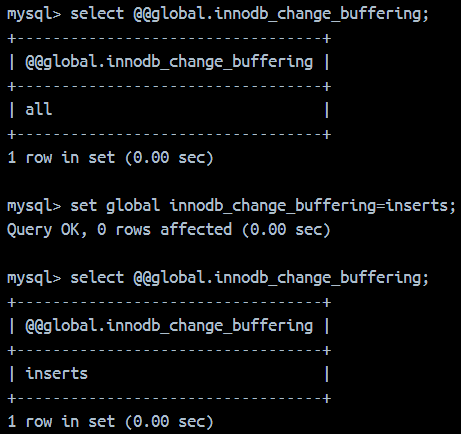
Chage Buffer u memoriji egzistira kao deo Buffer Pool-a, što je ilustrovano arhitekturom InnoDB-a na slici 4. Podrazumevano zauzima 25% memorije koju zauzima Buffer Pool, a maksimalno može da zauzima 50% Buffer Pool-a. Ovaj procenat se može podesiti manuelno postavljanjem ***innodb\_change\_buffer\_max\_size*** sistemske promenljive na željenu vrednost (na slici 5 je prikazano postavljanje kapaciteta Change Buffer-a na 40%). Ukoliko naša baza ne dobija puno zahteva za manipulacijom podataka, bilo bi dobro da se smanji kapacitet Change Buffer-a – u slučaju da korisnici izvršavaju veliki broj DML naredbi, performanse aplikacije bi značajno porasle povećanjem njegovog kapaciteta.

  
Slika 5. Postavljanje kapaciteta Change Buffer-a

Change Buffer se podešavanjem sistemske promenljive ***innodb\_change\_buffering*** može podesiti tako da kešira samo određene naredbe. Vrednosti koje može imati ova promenljiva su:

* *all* – baferovanje svih naredbi, odnosno INSERT, DELETE i *purge*. Purge naredba je slična DELETE naredbi, s tim što ona oslobađa prostor na disku koju je zauzimao obrisani podatak, dok DELETE naredba samo označi podatak kao obrisan, a i dalje ga čuva na disku,
* *none* – ne baferuje nikakve naredbe,
* *inserts,*
* *deletes,*
* *changes* – baferovanje naredbi INSERT i DELETE. Važno je napomenuti da naredba UPDATE predstavlja kombinaciju naredbi INSERT i DELETE,
* *purges.*

Na slici 6 je prikazana promena režima rada Change Buffer-a.

  
Slika 6. Postavljanje režima rada Change Buffer-a

U slučaju ažuriranja indeksirane kolone, baferovanje je moguće samo ukoliko je u pitanju sekundarni indeks. Sekundarni indeks je indeks koji se ne koristi za jedinstvenu identifikaciju podatka, ne skladišti podatke na disku u redu u kome pokazuje na njih, tako da promena te kolone na zahteva promenu redosleda podataka na disku.

### Adaptive Hash Index

Ova komponenta ima za cilj da ubrza izvršenje upita jer omogućava pronalazak direktne lokacije podatka u Buffer Pool-u na osnovu heširanog indeksa. InnoDB kreira Hash Index automatski za stranice indeksa kojima se često pristupa ukoliko uoči da bi upiti imali koristi od toga. Hash Index se gradi na osnovu prefiksa ili celog indeksnog ključa [7].

### Log Buffer

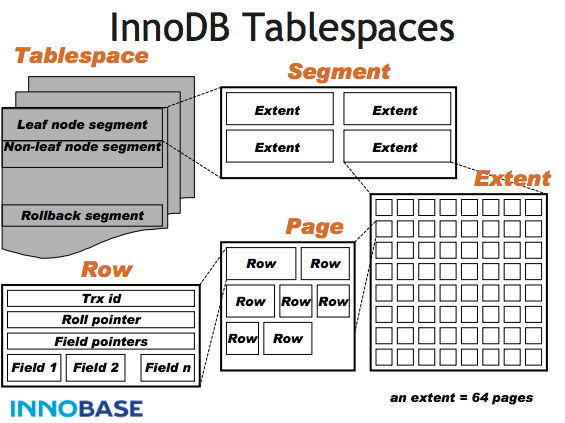
Log Bufer predstavlja oblast u memoriji koja baferuje logove transakcija koji trebaju biti upisani na disk. Veličina Log Buffer-a je definisana sistemskom promenljivom ***innodb\_log\_buffer\_size***, čija je podrazumevana vrednost 16MB. Ova In-Memory komponenta je usko povezana sa Redo Log komponentom koja se nalazi na disku, što se može videti sa slike 4 arhitekture InnoDB-a. Povećanjem kapaciteta Log Buffer-a (prikazano na slici 7) se omogućava izvršavanje transakcija koje se sastoje iz velikog broja naredbi bez potrebe da se logovi transakcije upisuju na disk zbog nedovoljnog kapaciteta Log Buffer-a. Time se smanjuje potreba za UI operacijama nad diskom za potrebe čuvanja logova.

  
Slika 7. Povećanje kapaciteta Log Buffer-a

## On-Disk komponente

Sa arhitekture prikazane na slici 4 se može videti da postoji 7 komponenti InnoDB-a koje služe za perzistenciju podatka na disku – od toga 5 *Tablespace* (srp. *prostor tabele*) komponenti, kao i *Redo Log* i *Doublewrite Buffer Files* (srp. *bafer dvostrukog upisa*) komponente.

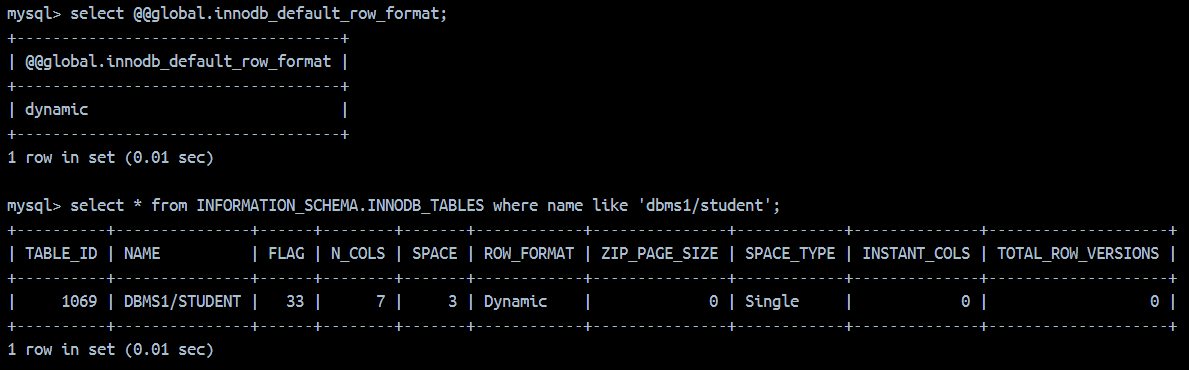
Tablespace predstavlja logički pogled na oblast na disku gde se smeštaju podaci. Tablespace-evi su podeljeni tako da je svaki zadužen za čuvanje određenih podataka. Najmanja jedinica Tablespace-a je stranica baze podataka koja je iste veličine za sve Tablespace-eve u sistemu. U stranici se čuvaju sami slogovi određene tabele. Stranica mora imati minimalno dva sloga. Podrazumevana veličina stranice je 16KB, dok se promenom vrednosti ***innodb\_page\_size*** sistemske promenljive može podesiti da bude 4KB, 8KB, 32KB ili čak 64KB. Važno je napomenuti da se promena ove promenljive, ukoliko želimo da veličina stranice ne bude podrazumevana, treba izvršiti pre pokretanja instance MySQL-a izmenom konfiguracionog fajla. Stranice su grupisane u oblasti (engl. *extent*) koji su veličine 1MB za stranice do 16KB, 2MB za stranice od 32KB, odnosno 4MB za stranice od 64KB. Koliko se stranica nalazi unutar jednog Extent-a se može izračunati ukoliko se veličina extenta podeli sa veličinom stranice[[2]](#footnote-2). Extent-ovi su dalje grupisani u segmente, koji čine jedan Tablespace [8]. Prethodno opisana organizacija Tablespace-a je prikazana na slici 8.

  
Slika 8. Organizacija Tablespace-a [9]

InnoDB omogućava da definišemo na koji način želimo da smeštamo slogove određene tabele na disku. Format reda [10] određuje kako su redovi fizički čuvaju na disku, što može uticati značajno na performanse upita i DML naredbi. Što više redova stane u jednu stranicu baze podataka, to će više njih biti učitano u radnu memoriju, odnosno u Buffer Pool. Uglavnom se pri definisanju tabele u bazi podataka definiše kog su tipa određene kolone, kako bi InnoDB znao koliko mesta da rezerviše za svaku kolonu. Problem nastaje kod kolona koje nisu fiksne, već promenljive dužine, kao što su VARCHAR, VARBINARY, BLOB i TEXT. Razlikujemo četiri tipa formata, a to su:

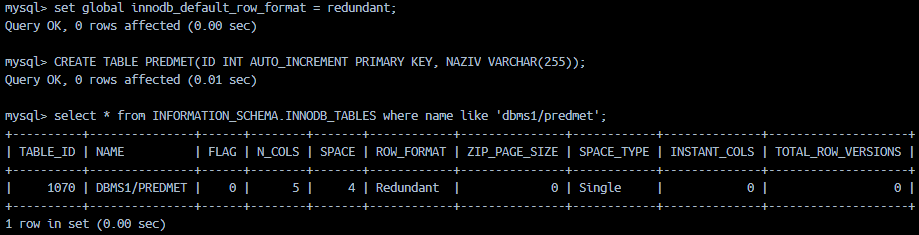
* REDUNDANT – ovaj format je kompatibilan sa starijim verzijama MySQL-a. Prvih 768 bajtova kolone koja predstavlja promenljivi tip podataka smešta unutar samog sloga, dok ostale bajtove skladišti na posebnoj stranici. Isto važi i za podatke fiksne veličine koje u zavisnosti od kodiranja mogu premašiti 768 bajtova, kao na primer CHAR(300) – ukoliko se za čuvanje karaktera koriste 1 ili 2 bajta, ceo podatak kolone će stati unutar jedne stranice, u suprotnom će ostatak morati da se smesti na posebnoj stranici.
* COMPACT – smanjuje zauzetost sloga na disku u odnosu na REDUDANT format za oko 20%. Kao i kod REDUNDANT formata važi da se prvih 768 bajtova podatka kolone skladište u stranici zajedno sa ostatkom sloga, dok se preostali deo smešta u okviru druge stranice.
* DYNAMIC – ovo je podrazumevani tip formata u MySQL verziji 8.3. Istih je karakteristika kao i COMPACT, s tim što kolone koje su promenljive dužine, ili fiksne dužine koja premašuje 768 bajtova, smešta u potpunosti van stranice na kojoj se nalaze ostali podaci sloga. Kako bi mogao da referencira te podatke, koristi pokazivač od 20 bajtova.
* COMPRESSED – istih je karakteristika kao DYNAMIC format, s tim što nudi podršku za kompresovanje podataka.

Sistemska promenljiva ***innodb\_default\_row\_format*** služi da definiše koji je tip formata reda se koristi trenutno. Podrazumevani tip je DYNAMIC, i sve tabele koje su kreirane pre promene vrednosti ***innodb\_default\_row\_format*** promenljive koriste ovaj tip za smeštanje podataka (prikazano na slici 9).

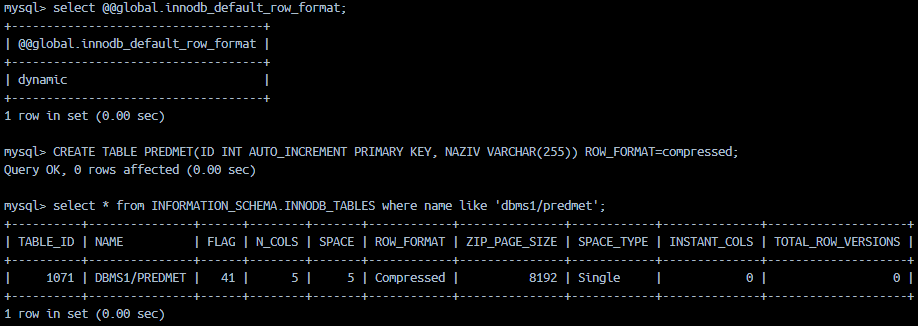
  
Slika 9. Prikaz podrazumevanog tipa formata

Promena tipa formata je moguća na dva načina:

1. da se promeni vrednost sistemske promenljive, čime sve novokreirane tabele preuzimaju novi format (prikazano na slici 10).

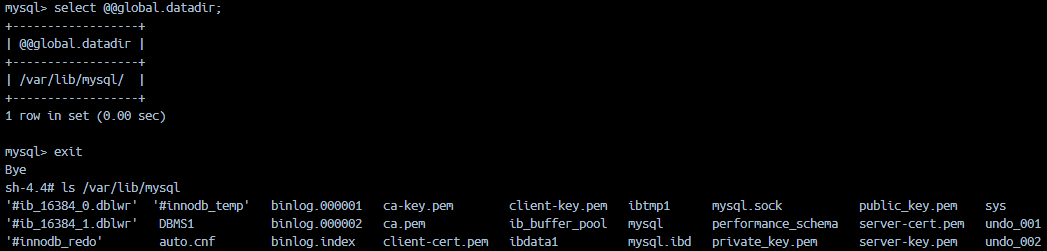
  
Slika 10. Promena tipa formata ažuriranjem sistemske promenljive

1. da se prilikom kreiranje tabele, nezavisno od vrednosti sistemske promenjlive, definiše koji tip formata će da se koristi za smeštanje njenih podataka (prikazano na slici 11).

  
Slika 11. Definisanje formata prilikom kreiranja tabele

### System Tablespace

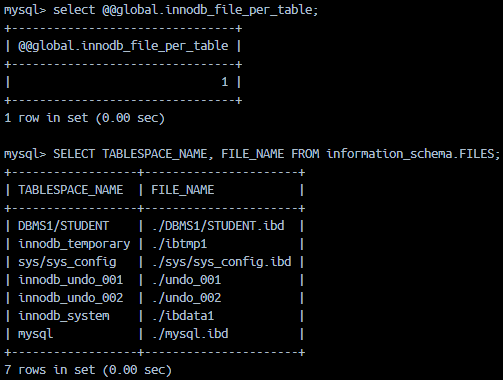
*System Tablespace* predstavlja oblast na disku koju InnoDB koristi za čuvanje različitih konfiguracija baze, odnosno metapodataka, Change Buffer-a i Doublewrite Buffer-a [11]. Takođe može čuvati podatke određenih relacija ukoliko korisnik tako zahteva. Ovo je jedini Tablespace koji može imati više od jednog fajla na disku. Prilikom kreiranja baze kreira se jedan ***ibdata1*** fajl u direktorijumu koji InnoDB koristi za smeštanje podataka (na slici 12 se vidi na koj direktorijum ukazuje sistemska promenljiva ***datadir*** koja označava gde InnoDB, odnosno MySQL skladišti podatke). Ovim fajlom su predstavljeni podaci koje skladišti System Tablespace. Zanimljivo je da je promena veličine System Tablespace-a moguća samo ukoliko korisnik želi da poveća njegov kapacitet, a ne i kad želi da ga smanji.

  
Slika 12. Direktorijum gde InnoDB skladišti podatke i prikaz ***ibdata1*** fajla prilikom listanja sadržaja direktorijuma koji se koristi za skladište

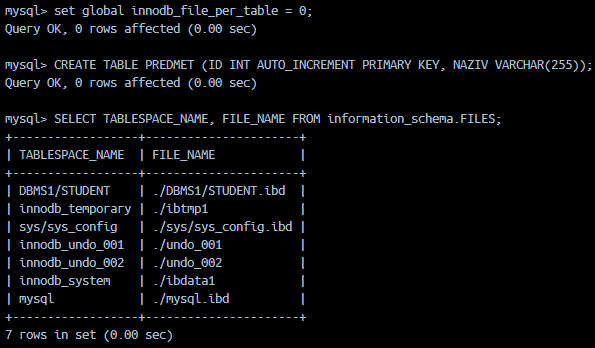
### File-Per-Table Tablespace

*File-Per-Table* (skraćeno FPT) *Tablespace*-eva [12] može biti više u sistemu. Razlog tome je što se kreira poseban Tablespace ovog tipa za svaku tabelu koja se kreira u bazi, i na disku se čuva u okviru jednog fajla.

Korisnik može da podešava da li želi da se kreira poseban Tablespace za svaku tabelu u bazi konfigurisanjem sistemske promenljive ***innodb\_file\_per\_table***. Koja može uzimati vrednosti 0 i 1. Ukoliko želimo da se kreira poseban Tablespace za svaku tabelu, treba je postaviti na 1, u suprotnom na 0. Kada je postavljena na 0, podaci tabele i njeni metapodaci, kao i indeksi tabele će se čuvati u okviru System Tablespace-a. Podrazumevano je omogućeno kreiranje Tablespace-a za svaku tabelu.

  
Slika 13. Podrazumevano podešavanje FPT Tablespace-a i prikaz kreiranja zasebnog fajla za tabelu STUDENT

Na slici 13 je prikazano da je podrazumevano omogućen FPT Tablespace i prikazan je fajl (*./DBMS1/STUDENT.idb*) koji čuva podatke za STUDENT tabelu u okviru DBMS1 baze podataka, dok je na slici 14 se može videti da se onemogućavanjem ***innodb\_file\_per\_table*** sistemske promenljive ne kreira poseban Tablespace za novokreiranu tabelu.

  
Slika 14. Rezultat postavljanja ***innodb\_file\_per\_table*** promenljive na 0

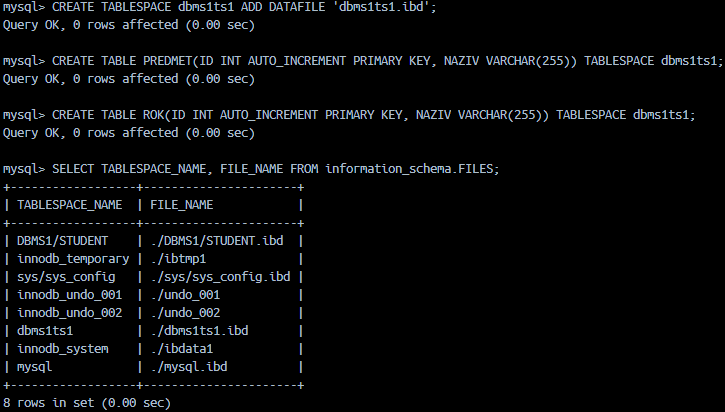
Dobra stvar kreiranja posebnog Tablespace-a za svaku tabelu je to što se prilikom izbacivanja tabele iz baze podataka operativnom sistemu vraća prostor koji je ona zauzimala. Ukoliko se iz baze izbaci tabela koja se nalazila u okviru deljivog Tablespace-a, tada se prostor koji je ona zauzimala vraća Tablespace-u na korišćenje, a ne operativnom sistemu. Mana toga je mala verovatnoća da će se taj prostor ponovo iskoristiti. Još jedna prednost posebnog Tablespace-a za svaku tabelu je to što je moguće da se tabele kreiraju na nekom eksternom disku, čime se omogućava da se ta tabela koristi u nekoj drugoj bazi podataka.

### General Tablespace

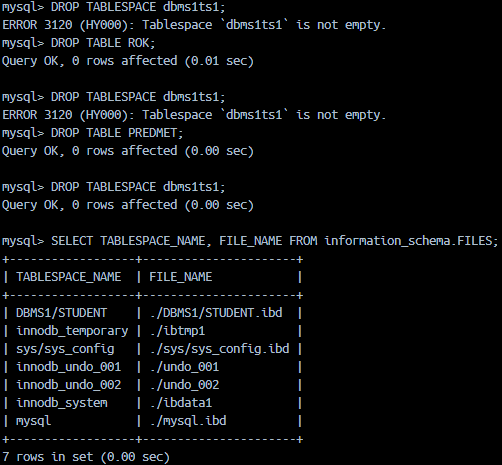
*General Tablespace* [13] je deljivi Tablespace koji korisnici, za razliku od prethodna dva opisana Tablespace-a, moraju eksplicitno da kreiraju CREATE TABLESPACE naredbom. Predstavlja Tablespace koji mogu koristiti više tabela.

Prednost u odnosu na FPT Tablespace-eve je ta što sistem metapodatke General Tablespace-a drži sve vreme u radnoj memoriji dok Tablespace postoji.

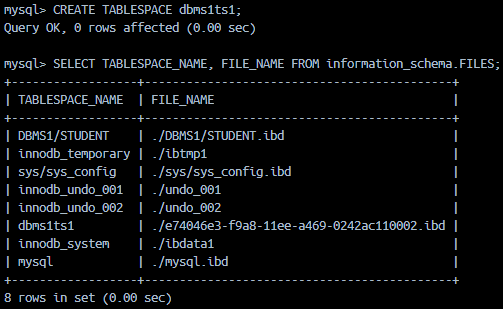
Ukoliko želimo da se naša tabela skladišti unutar nekog General Tablespace-a, neophodno je da se prilikom CREATE TABLE naredbe navede kom Tablespace-u će tabela pripadati. Na slici 15 je prikazano kreiranje General Tablespace-a ***dbms1ts1*** i njegovo dodeljivanje PREDMET tabeli. Takođe se može videti da se novokreirani Tablespace prikazuje u tabeli Tablespace-eva. Uočavamo da je dvema tabelama dodeljen ***dbms1ts1*** Tablespace, čime je dokazano da više tabela može deliti jedan General Tablespace.

  
Slika 15. Kreiranje novog General Tablespace-a i njegovo dodeljivanje tabelama

General Tablespace se, analogno njegovom kreiranju, mora eksplicitno obrisati ukoliko želimo da ga uklonimo iz sistema DROP TABLESPACE naredbom. Ono o čemu moramo voditi računa je da Tablespace mora biti prazan kada želimo da ga obrišemo, odnosno da ne sadrži nijednu tabelu, što je ilustrovano na slici 16.

  
Slika 16. Brisanje General Tablespace-a

Može se primetiti da je na slici 15 prilikom kreiranja Tablespace-a naveden i fajl koji će se koristiti za njega. Ukoliko se ne navede naziv fajla, InnoDB će sam kreirati jedinstveni naziv za fajl koji se sastoji iz 128 bitova formatiranih u 5 grupa koje su razdvojene crticama (prikazano na slici 17).

  
Slika 17. Kreiranje Tablespace-a bez navođenja imena fajla

### Undo Tablespace

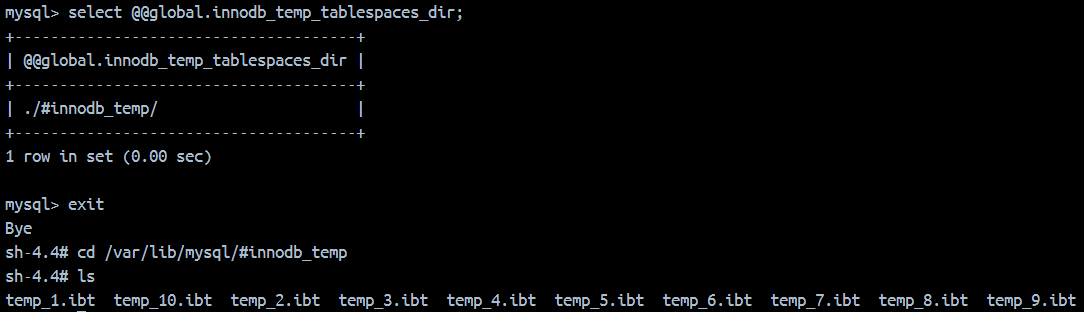
Ovaj Tablespace se koristi da čuva Undo logove, koji sadrže informacije kako da se ponišite poslednje promene transakcije koja nije komitovana [14]. Ukoliko neka transakcija zahteva čitanje podataka koja je druga transackija izmenila, iz Undo logova se generiše stanje podataka koje je bilo pre njihove modifikacije.

Prilikom pokretanja instance MySQL-a se podrazumevano kreiraju dva Undo Tablespace-a kako bi, u slučaju da se jedan popuni, drugi mogao da nastavi da prihvata logove koji nastaju kao posledica izvršenja transakcija.

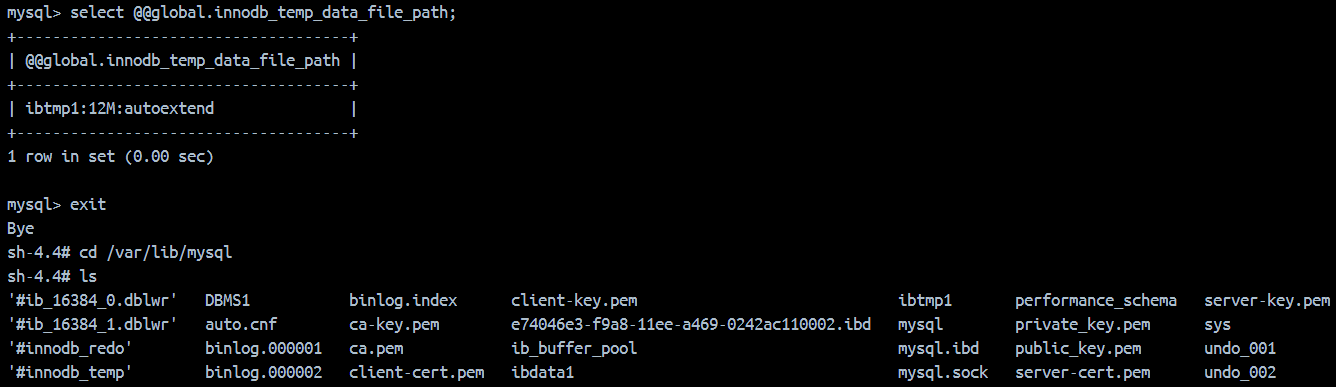
### Temporary Tablespace

*Temporary* (srp. *privremeni*) *Tablespace* [15] može biti:

* Session Temporary Tablespace – koristi se za čuvanje privremenih tabela koje je kreirao korisnik u okviru jedne sesije. One se alociraju iz opsega (engl. *pool*) privremenih Tablespace-eva. Prilikom startovanja MySQL instance, u opsegu se nalazi njih deset. Promenljiva ***innodb\_temp\_tablespace\_dir*** ukazuje na lokaciju gde se ovi Tablespace-evi nalaze, što je prikazano na slici 18. Zanimljivo je primetiti da ovi Tablespace-evi, za razliku od ostalih, koriste ekstenziju *.ibt*.

  
Slika 18. Prikaz Session Temporary Tablespace-eva

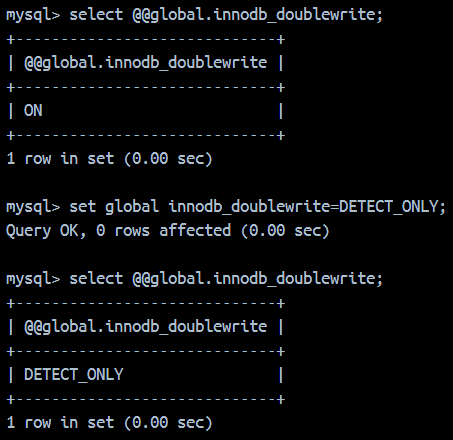
* Global Temporary Tablespace – ovaj tablespace se koristi za čuvanje *rollback* segmenata koji nastaju kao rezultat izvršenja transakcija nad privremenim tabelama korisnika. Promenljiva ***innodb\_temp\_data\_file\_path*** definiše relativnu putanju, veličinu i atribute ovog Tablespace-a (prikazano na slici 19). Atribut *autoextend* znači da se po potrebi veličina ovog Tablespace-a može automatski povećati ukoliko za tim bude bilo potrebe.

  
Slika 19. Atributi i lokacija Global Temporary Tablespace-a

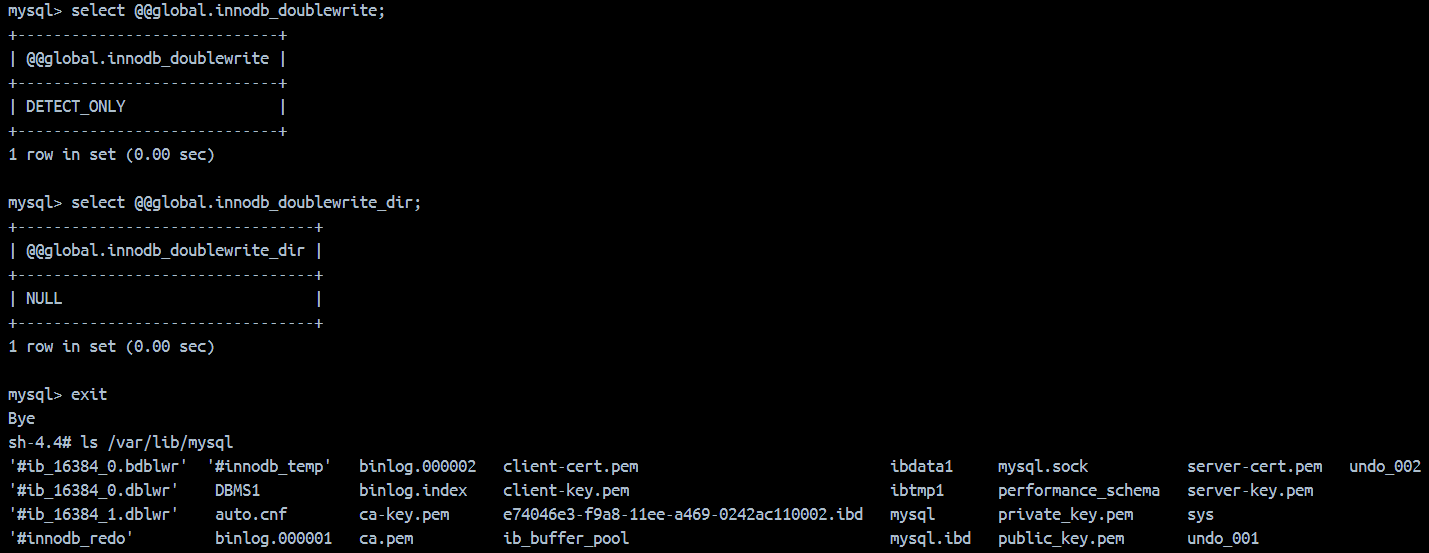
### Doublewrite Buffer

U trenutku upisa modifikovanih stranica iz Buffer Pool-a na disk, može doći do iznenadnog kraha sistema, što bi rezultovalo gubitkom podataka, jer se modifikovani podaci iz radne memorije nakon gašenja sistema više ne nalaze u njoj, a nisu upisani na perzistentni medijum. Uloga bafera dvostrukog upisa je da se prevaziđu ovakve situacije, time što se modifikovane stranice iz memorije prvo upišu u njega, a zatim se prepisuju u odgovarajuće lokacije na disku [16]. Iako se ovim podaci upisuju dva puta na disk, to ne znači da se zahteva dvostruko više UI operacija. Stranice koje se upisuju u Doublewrite Buffer se upisuju kao velika sekvenca podataka jednim pozivom operativnog sistema za upis na disk. Ukoliko dođe do iznenadnog prekida rada sistema prilikom upisa modifikovanih stranica, prilikom ponovnog pokretanja instance MySQL-a će se stanje izgubljenih podataka oporaviti iz Doublewrite Buffer-a.

Sistemska promenljiva ***innodb\_doublewrite*** ukazuje da li je korišćenje Doublewrite Buffer-a omogućeno ili ne. Podrazumevano je omogućen. Pored ON i OFF vrednosti, pri čemu se ON vrednost može predstaviti i kao DETECT\_AND\_RECOVER, moguće je da se odabere DETECT\_ONLY režim rada Doublewrite Buffer-a u slučaju kada želimo da se u njega upisuju samo metapodaci o modifikovanim stranicama kako bismo znali koje su stranice modifikovane, a ne i kako. Postavljanje režima rada Doublewrite buffer-a je prikazano na slici 20.

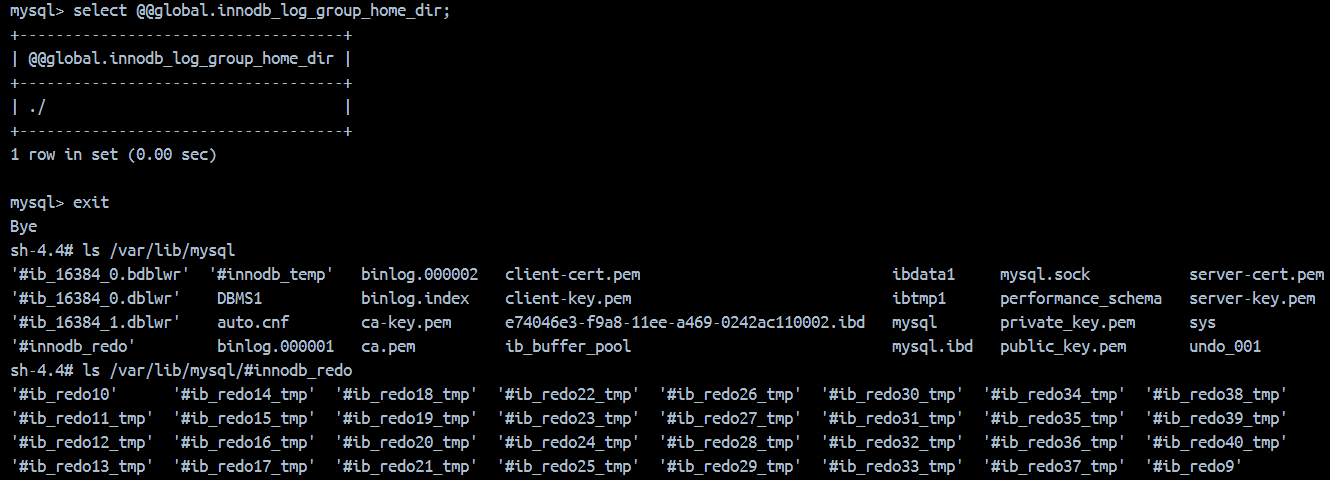
  
Slika 20. Prikaz i modifikacija režima rada Doublewrite Buffer-a

Promenljiva ***innodb\_doublewrite\_dir*** ukazuje na direktorijum u kome se smeštaju Doublewrite Buffer fajlovi, dok ***innodb\_doublewrite\_files*** ukazuje na to koliko će fajlova koristiti Doublewrite Buffer. Ukoliko nije specificiran direktorijum za čuvanje Doublewrite Buffer fajlova, fajlovi se smeštaju u direktorijum koji je definisan ***datadir*** promenljivom, što je prikazano na slici 21. Fajlovi koje koristi Doublewrite Buffer su formata *#ib\_brojStranice\_****broj****.dblwr* (ili .bdblwr ukoliko se koristi DETECT\_ONLY režim rada). InnoDB podrazumevano krerira dva fajla za potrebe Doublewrite Buffer-a.

  
Slika 21. Perzistencija Doublewrite Buffer fajlova u fajl sistemu

### Redo Log

Kao i Doublewrite Buffer, Redo Log komponenta [17] ima ulogu u oporavku stanja baze podataka prilikom iznenadnog pada sistema, ali u drugom kontekstu. Dok Doublewrite Buffer ima za cilj oporavak podataka koji su trebali da budu upisani na disk, a usled nepredviđenih okolnosti nisu, Redo Log ima ulogu da oporavi stanje transakcija koje su se još uvek izvršavale u trenutku iznenadnog prekida rada sistema, kako korisnik ne bi morao da pamti koje je akcije sve izvršio u toku trajanja transakcije. Ideja je da se u Redo Log-u pamti koji su podaci izmenjeni i na koji način, kako bi prilikom oporavka baze podataka moglo da se oporavi stanje transakcija izvršenjem naredbi iz Redo Log-a nad originalnim podacima sa diska. Nakon toga se korisniku prikazuju podaci u onom obliku u kome su bili u toku trajanja transakcije pre nastanka kraha.

  
Slika 22. Prikaz Redo log fajlova na fajl sistemu

Promenljiva ***innodb\_redo\_log\_group\_home\_dir*** pokazuje na direktorijum (na slici 22 se vidi da ova promenljiva ukazuje na direktorijum koji koristi MySQL za skladištenje podataka, što je */var/lib/mysql*) u kome se nalazi *#innodb\_redo* direktorijum u kome se nalaze Redo Log fajlovi. Redo Log fajlovi mogu biti obični (engl. *ordinary*) i rezervni (engl. *spare*). Razlikuju se u tome što su Orindary fajlovi (koji se ne završavaju sa *\_tmp*) trenutno u upotrebi, dok se Spare fajlovi (završavaju se sa *\_tmp*) čuvaju u rezervi za buduće potrebe. Broj Redo Log fajlova koje održava InnoDB je 32, pri čemu je konvencija imenovanja fajlova *#ib\_redo****N*** , gde N označava redni broj fajla.

# Zaključak

U radu je prvo opisana uloga sistema za upravljanje bazom podataka i njegova arhitektura na primeru MySQL-a. Nakon toga je predstavljena uloga Storage Engine-a te arhitekture, pri čemu je opisan InnoDB kao podrazumevani Storage Engine za MySQL.

InnoDB se sastoji iz dve grupe komponenti: onih koje funkcionišu u organizaciji podataka u radnoj memoriji i onih koje učestvuju u perzistenciji podataka i metapodataka. Uloga svake komponente je ukratko opisana i ilustrovana primerima koji su imali za cilj da pokažu lokacije na disku gde InnoDB čuva podatke, kao i upravljanje samim InnoDB Storage Engine-om. O skoro svakoj komponenti InnoDB-a bi mogao da se napiše poseban rad, čime se pokazuje koliko je kompleksno funkcionisanje sistema zaduženog za skladištenje podataka.

# Literatura

[1] Difference Between DBMS and RDBMS, dostupno na: <https://www.kaashivinfotech.com/blog/difference-between-dbms-and-rdbms/> (pristupljeno 8. aprila 2024.)

[2] Overview of MySQL Storage Engine Architecture, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/pluggable-storage-overview.html> (pristupljeno 8. aprila 2024.)

[3] MySQL 8.3 Supported Storage Engines, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/storage-engines.html> (pristupljeno 8. aprila 2024.)

[4] InnoDB Architecture, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/innodb-architecture.html> (pristupljeno 11. aprila 2024.)

[5] Buffer Pool, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/innodb-buffer-pool.html> (pristupljeno 11. aprila 2024.)

[6] Change Buffer, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/innodb-change-buffer.html> (pristupljeno 11. aprila 2024.)

[6] Adaptive Hash Index, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/innodb-adaptive-hash.html> (pristupljeno 12. aprila 2024.)

[7] Log Buffer, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/innodb-redo-log-buffer.html> (pristupljeno 12. aprila 2024.)

[8] File Space Management, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/innodb-file-space.html> (pristupljeno 12. aprila 2024.)

[9] InnoDB Tablespaces, dostupno na: <https://jackyfkc.github.io/cs/data_store/mysql/innodb.html> (pristupljeno 12. aprila 2024.)

[10] InnoDB Row Formats, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/innodb-row-format.html> (pristupljeno 13. aprila 2024.)

[11] The System Tablespace, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/innodb-system-tablespace.html> (pristupljeno 13. aprila 2024.)

[12] File-Per-Table Tablespaces, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/innodb-file-per-table-tablespaces.html> (pristupljeno 13. aprila 2024.)

[13] General Tablespaces, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/general-tablespaces.html> (pristupljeno 13. aprila 2024.)

[14] Undo Tablespaces, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/innodb-undo-tablespaces.html> (pristupljeno 13. aprila 2024.)

[15] Temporary Tablespaces, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/innodb-temporary-tablespace.html> (pristupljeno 13. aprila 2024.)

[16] Doublewrite Buffer, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/innodb-doublewrite-buffer.html> (pristupljeno 14. aprila 2024.)

[17] Redo Log, dostupno na: <https://dev.mysql.com/doc/refman/8.3/en/innodb-redo-log.html> (pristupljeno 14. aprila 2024.)

1. Slično */dev/null* fajlu u Linux operativnom sistemu [↑](#footnote-ref-1)
2. Na primer, za stranice veličine 32KB, Extent je 2MB. Ukoliko podelimo 2MB sa 32KB dobićemo 64 stranice u Extent-u. [↑](#footnote-ref-2)