Indeksiranje v SQL

- Indeks je za uporabnika nevidna podatkovna struktura, ki bistveno pospeši dostop do vrstic tabele; preiskovanje *n* vrstic: s $t_1=O(n)$ na $t_2=O(\log n)$;
- pri n=1 milijon je $t_1= 1000000$, $t_2= 6$ (stevilo korakov)
- Indeksiramo po enem ali več stolpcih
- Zakaj vedno ne indeksiramo celotne tabele:
  - za *k* atributov je možnih 2^k indeksov (vse podmnožice)
  - vsak indeks zahteva prostor na disku in čas za njegovo gradnjo in posodabljanje ob spremembah tabele

Kdaj zgraditi indeks na podmnožici atributov?

- Ključi in ostali enolični (UNIQUE) atributi: pogosto avtomatsko generiranje
- Pogostost preiskovanja in urejanja
- Velikost tabele
- Porazdelitev podatkov
- Prostorsko-časovne omejitve: prostor na voljo v PB, pogostost spreminjanja tabele

Kreiranje in brisanje indeksov

- Kreiranje indeksov:

```
CREATE [UNIQUE] INDEX ime_indeksa
ON ime_tabele (ime_atributa1 [ASC|DESC],
               ime_atributa2 [ASC|DESC],
               ... );
```

- Indeks se gradi po kombinaciji vrednosti atributov; za vsako kombinacijo atributov potrebujemo svoj indeks
- Možna specifikacija tipa indeksa (npr. BTREE, HASH)
- Brisanje nepotrebnih indeksov:

```
DROP INDEX ime_indeksa ON ime_tabele;
```

Primer indeksiranja

- Indeksiraj čolne po barvi!

```
CREATE INDEX po_barvi
ON coln(barva);
```

- Indeksiraj rezervacije ločeno po datumih ter šifrah jadralcev in čolnov skupaj!

```
CREATE INDEX po_dnevih
ON rezervacija(dan);

CREATE INDEX po_jid_cid
ON rezervacija(jid,cid);
```

Uporaba indeksov

- Indeksi se uporabljajo avtomatsko, ko jih enkrat kreiramo; sistem izbere, katerega od potencialno več možnih obstoječih bo uporabil
- Eksplicitna (ne)uporaba indeksov: dosežemo s specialnimi komentarji ali ukazi - namigi (hints)
- Zakaj namigi? Ker vnaprej vemo več kot sistem o tem, kako se bodo podatki uporabljali
- Namigi so NESTANDARDNA razširitev SQL

Namigi za indeksiranje v MariaDB

- Namig: dodana ključna beseda v SELECT stavku za imenom tabele v FROM vrstici

```
-- Uporabi samo naštete indekse
USE INDEX(ime_indeksa1, ime_indeksa2, ...)

-- Ne uporabi nobenega indeksa
USE INDEX()

-- Ignoriraj naštete indekse
IGNORE INDEX(ime_indeksa1, ime_indeksa2, ...)
```

Primeri nekaterih namigov v MariaDB

- Denimo, da smo vnaprej kreirali indekse `jad_index1(jid,ime), jad_index2(jid), jad_index3(ime).`

```
SELECT *
FROM jadralec
   USE INDEX(jad_index1)      -- uporabi indeks
ORDER BY ime, jid;            -- po jid in imenu

SELECT *
FROM jadralec
   IGNORE INDEX(jad_index1, jad_index2, jad_index3)
ORDER BY ime, jid;  -- ne uporabi nobenega nastetega
                    -- indeksa
```

Merjenje časa v MariaDB

Z uporabo profilinga lahko izmerite čas izvajanja poizvedb, da ugotivite kateri index je najbolj primeren.

```
SET profiling = 1;
SELECT * FROM …;
-- pokaže pretekle poizvedbe in čas izvajanja
SHOW PROFILES;
-- pokaže podrobnosti poizvedbe 1 (iz prejšnje tabele)
SHOW PROFILE FOR QUERY 1;
```

Procedure

- Shranjeni podprogrami, ki jih lahko kličemo
- Parametre lahko določimo kot vhodne (IN), izhodne (OUT) ali vhodno-izhodne (IN OUT)
- Kličemo jih s CALL Ime_Procedure(…)

Primer

Napišite shranjeno proceduro, ki vrne vse podatke o čolnih z dolžino med "spodnja" in "zgornja" meja.

```
DELIMITER //
CREATE PROCEDURE colni_razpon (IN spodnja
INTEGER, IN zgornja INTEGER)
BEGIN
        SELECT * FROM coln
        WHERE dolzina BETWEEN spodnja AND zgornja;
END //
DELIMITER ;

CALL colni_razpon(20,40);
```

Dostop do parametrov

- Naprimer, da imate proceduro rezervacijeJadralca(IN *ime* VARCHAR(255), OUT *n* INTEGER), ki vam v zadnjem parametru vrne število rezervacij jadralcev z imenom *ime*.

```
CALL rezervacijeJadralca(„Henrik“, @a);
SELECT @a;
```

Funkcije

- Podobno kot procedure, le da vračajo vrednost
- Parametri so privzeto IN
- Uporaba SELECT Ime_Funkcije(…)

Primer

Napišite funkcijo, ki vrne število čolnov z dolžino med "spodnja" in "zgornja" meja.

```
DELIMITER //
CREATE FUNCTION colni_razpon_fun (spodnja
INTEGER, zgornja INTEGER) RETURNS INTEGER
BEGIN
DECLARE X INTEGER;
SELECT COUNT(*) INTO X
FROM coln
WHERE dolzina BETWEEN spodnja AND zgornja;
       RETURN X;
END //
DELIMITER ;
SELECT colni_razpon_fun(20,40);
```

Dostop do rezultata

- Naprimer, da imate proceduro rezervacijeJadralca(IN *ime* VARCHAR(255)), ki vam vrne število rezervacij jadralcev z imenom *ime*.

SELECT rezervacijeJadralca(„modra“)

Brisanje procedur in funkcij

- DROP PROCEDURE IF EXISTS Ime_procedure;
- DROP FUNCTION IF EXISTS Ime_Funkcije;

Bazni prožilci

- Podobni proceduram, le da nimajo argumentov in se avtomatsko kličejo ob ažuriranju tabele.
- Lahko se kličejo pri vstavljanju (INSERT), brisanju (DELETE) ali posodabljanju (UPDATE).
- Stavčni in vrstični prožilci.

Vaja

- Kreirajte tabelo *MladoletniJadralci* s funkcionalnostjo materializiranega pogleda, ki hrani samo jadralce mlajše od 18 let. Dodajte bazne prožilce za vstavljanje, brisanje in posodabljanje vrstic originalne tabele *jadralci*.

Ustvarjanje tabele

```
CREATE TABLE MaldoletniJadralci AS
        SELECT *
        FROM jadralec
        WHERE starost <18;
```

Bazni prožilec za vstavljanje

```
DELIMITER //
CREATE TRIGGER MladoletniJadralci_Insert
AFTER INSERT ON jadralec
FOR EACH ROW
BEGIN
IF NEW.starost <18 THEN
  INSERT INTO MladoletniJadralci VALUES(NEW.jid, NEW.ime,
                                        NEW.starost, NEW.rating);
END IF;
END //
```

Vaje

1. Po tabeli rezervacij pogosto preiskujemo po atributih jid in cid posamezno, ter po (jid, cid) skupaj. Kreirajte ustrezne indekse!
2. Napišite proceduro, ki vam vrne vse čolne določene barve ter v zadnjem parametru vrne število le teh.
3. Prejšnja naloga, le da jo implementirajte kot funkcijo.
4. Tabeli MladoletniJadralci dodajte še bazne prožilce za brisanje in spreminjanje vrstic iz tabele jadralci.