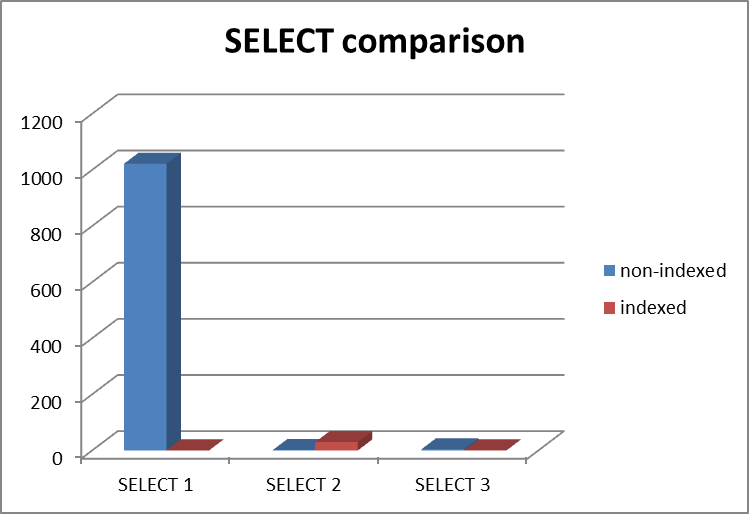
# MySQL indeksų tyrimas

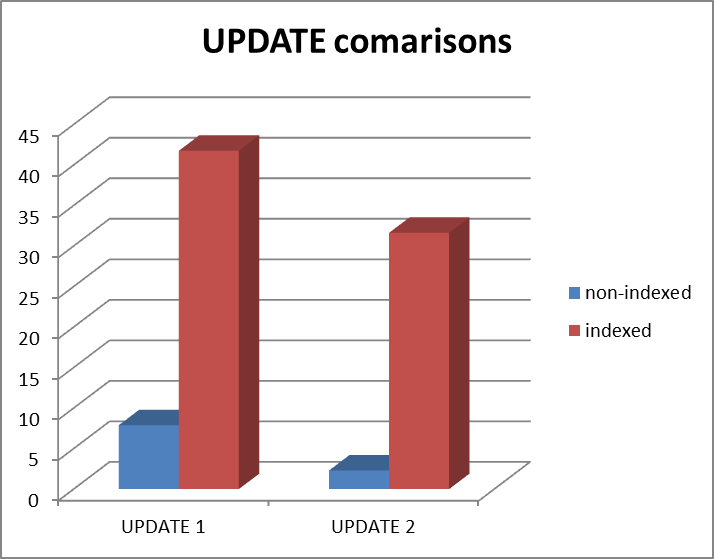
## SELECT

|  |  |  |  |
| --- | --- | --- | --- |
| Query | Name | Time took (non-indexed), s | Time took (indexed), s |
| SELECT SQL\_NO\_CACHE \*  FROM jobsregister as jr  LEFT JOIN jobsregister\_materials as jm ON jr.jobsRegisterId = jm.jobsREgisterId and jr.kkTechnicianId = 22  LIMIT 10; | SELECT 1 | 1021,97 | 0,01 |
| SELECT SQL\_NO\_CACHE jobsRegisterId as job\_id, SUM(mat.cost \* jm.count) as total\_cost  FROM `jobsregister\_materials` as jm  INNER JOIN `materials` as mat  ON mat.materialId = jm.materialId  GROUP BY jobsRegisterId  ORDER BY total\_cost DESC  LIMIT 5; | SELECT 2 | 1,45 | 30,05 |
| SELECT SQL\_NO\_CACHE serv.name, count(\*) as times\_registered  FROM `services` as serv  LEFT JOIN `jobsregister\_services` AS js ON js.serviceId = serv.serviceId  GROUP BY serv.name  ORDER BY times\_registered DESC  LIMIT 10; | SELECT 3 | 4,39 | 0,09 |



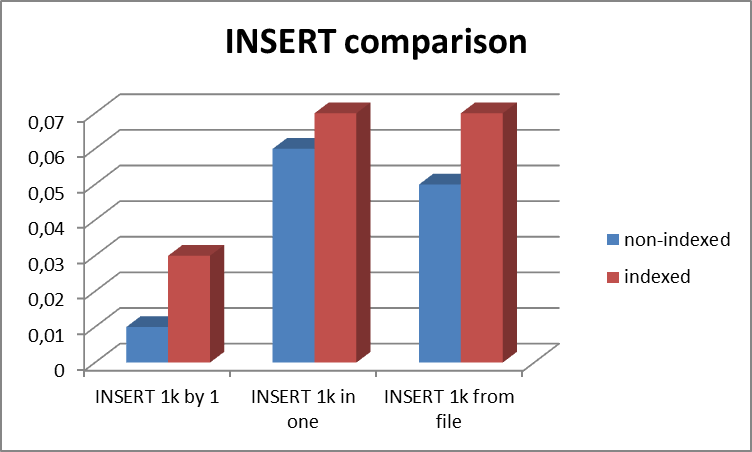
## UPDATE

|  |  |  |  |
| --- | --- | --- | --- |
| Query | Name | Time took (non-indexed), s | Time took (indexed), s |
| UPDATE jobsregister jobs INNER JOIN  (SELECT jobsRegisterId as job\_id, SUM(mat.cost) as total\_cost FROM `jobsregister\_materials` as jm INNER JOIN `materials` as mat ON jm.materialId = mat.materialId GROUP BY jobsRegisterId) as costs ON costs.job\_id = jobs.jobsRegisterId AND costs.total\_cost < 1000 AND costs.total\_cost != 0 SET jobs.type = 'not\_so\_expensive\_jobs'; | UPDATE 1 | 7,88 | 41,74 |
| UPDATE jobsregister jobs INNER JOIN  (SELECT jobsRegisterId as job\_id, SUM(jm.count) as total\_mats FROM `jobsregister\_materials` as jm INNER JOIN `materials` as mat ON jm.materialId = mat.materialId GROUP BY jobsRegisterId) as mats ON mats.job\_id = jobs.jobsRegisterId AND mats.total\_mats > 100 SET jobs.type = 'jobs\_with\_lot\_of\_mats'; | UPDATE 2 | 2,29 | 31,62 |



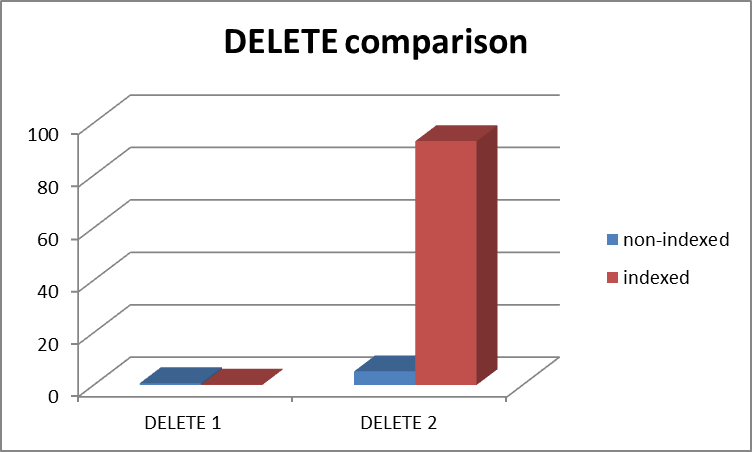
## INSERT

|  |  |  |  |
| --- | --- | --- | --- |
| Query | Name | Time took (non-indexed), s | Time took (indexed), s |
| Source 1k\_by\_one.txt | INSERT 1k by 1 | 0,01 | 0,03 |
| Source 1k\_in\_one.txt | INSERT 1k in one | 0,06 | 0,07 |
| LOAD DATA INFILE '1k\_in\_file.txt' INTO TABLE jobsregister\_materials | INSERT 1k from file | 0,05 | 0,07 |



## DELETE

|  |  |  |  |
| --- | --- | --- | --- |
| Query | Name | Time took (non-indexed), s | Time took (indexed), s |
| DELETE FROM jobsregister\_materials  WHERE jobsRegisterServiceId = 0 AND jobsRegisterId = 0 | DELETE 1 | 0,68 | 0,09 |
| DELETE  FROM jobsregister\_services WHERE goal = 'primary' | DELETE 2 | 5,19 | 93,03 |



## Išvados

Stulpelių, kurie yra išoriai raktai, indeksavimas gali žymiai pagerinti SELECT užklausos vykdymo laiką, ypač tų, kuriuose naudojamos JOIN operacijos. Sudėjus indeksus žymiai pagerėjo SELECT 1 ir SELECT 2 užklausos, bet pablogėja SELECT 3 užklausa (indeksas ant *jobsregister\_materials.materialId* ). Update užklausos taip pat pablogėjo (tikriausiai dėl to, kad abi UPDATE užklausos panašios į SELECT 2 užklausą). INSERT užklausos šiek tiek pablogėja su indeksais, bet dėl sąlyginai mažų duomenų kiekių (po 1000) laikų skirtumai yra per maži, kad galima būtų daryti objektyvias išvadas . NedidelėDELETE užklausa (DELETE 1), kur naudojamos indeksuojamos reikšmės pagerėjo, bet žymiai pablogėjo didelė DELETE užklausa (DELETE 2) lentelėje kurioje naudojami indeksai, bet pačioje užklausoje indeksuotos reikšmės nenaudojamos.