

II2250 Manajemen Basis Data
Semester II 2022/2023

Praktikum II: Indexing & SQL Tuning

Dipersiapkan oleh:
Asisten Laboratorium Basis Data

Pelaksanaan:
Waktu Mulai: Senin, 27 Februari 2023 – 16.15 WIB
Waktu Selesai: Senin, 27 Februari 2023 – 18.00 WIB

I. Ketentuan

Peserta kuliah akan mengerjakan praktikum mengenai **Schema Tuning dan Indexing** pada DBMS PostgreSQL menggunakan data hospital.sql yang tersedia di folder praktikum ini. Praktikum ini dikerjakan secara **INDIVIDU**.

Peserta dapat memanfaatkan file yang terdapat pada folder [Untuk Peserta](#), dokumentasi [PostgreSQL](#), dan sumber lain di internet dalam pengerjaan soal praktikum.

Deliverables yang harus dikumpulkan untuk Praktikum ini adalah sebagai berikut.

1. File dump database dengan format nama: **K<Kelas>_P02_<NIM>.sql**
Contoh: K01_P02_18221000.sql
2. File .docx berisikan daftar query beserta screenshot eksekusi tiap query dengan format nama: **K<Kelas>_P02_<NIM>.docx**
Contoh: K01_P02_18221000.docx

Kedua file dijadikan satu dalam format zip, dengan nama: **K<Kelas>_P02_<NIM>.zip**

Contoh: K01_P02_18221000.zip

Isi dari dokumen adalah sebagai berikut.

- a. Untuk setiap soal.
 - i. Perintah atau langkah yang dilakukan untuk menyelesaikan persoalan. Sediakan semua kode dalam bentuk **teks**.
 - ii. Query pengecekan dalam bentuk **teks** (jika diminta pada soal).
 - iii. Hasil perintah dari *query* tersebut

Deliverables dikumpulkan pada pranala berikut.

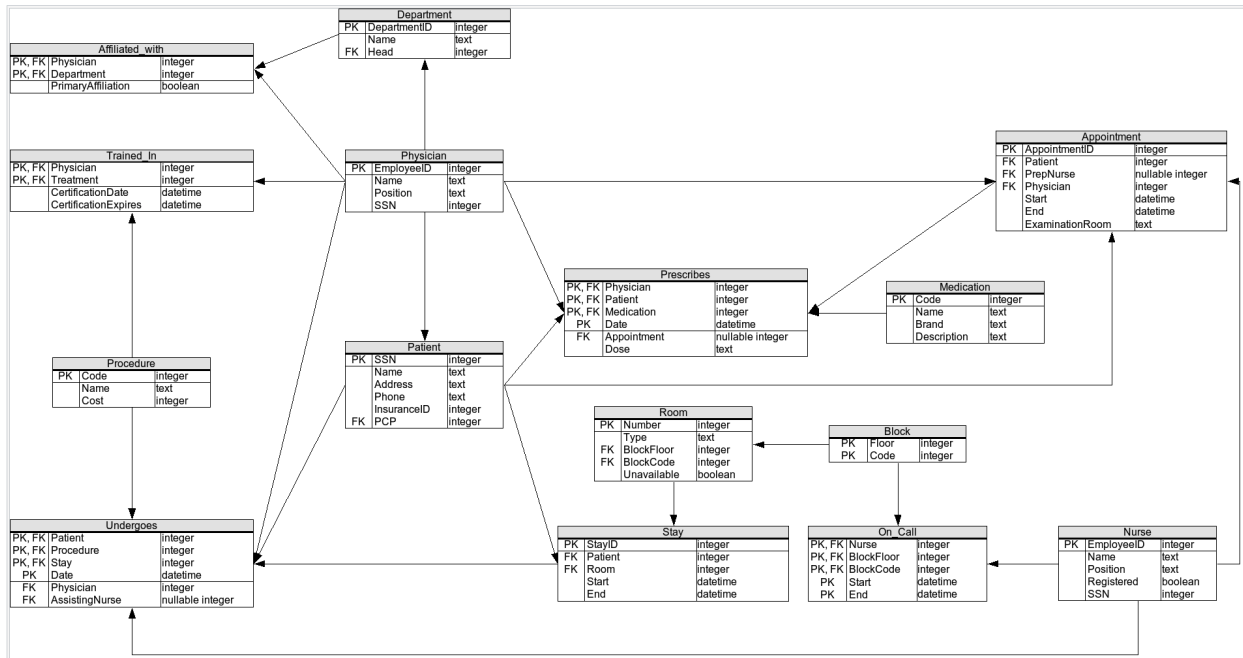
<https://bit.ly/prak2-uploader>

Keterlambatan pengumpulan *deliverables* akan mengakibatkan pengurangan nilai. Segala tindak kecurangan akan ditindaklanjuti dan diikuti konsekuensi serius.

~ Selamat mengerjakan! :)) ~

II. Skema Basis Data

Berikut ini adalah skema dari basis data Rumah Sakit Basdat dari file hospital.sql.



Affiliated_with = (Physician, Department, PrimaryAffiliation)

Appointment = (AppointmentID, Patient, PrepNurse, Physician, StartDate, EndDate, ExaminationRoom)

Block = (BlockFloor, BlockCode)

Department = (DepartmentID, Name, Head)

Medication = (Code, Name, Brand, Description)

Nurse = (EmployeeID, Name, Position, Registered, SSN)

On_call = (Nurse, BlockFloor, BlockCode, OnCallStart, OnCallEnd)

Patient = (SSN, Name, Address, Phone, InsuranceID, PCP)

Physician = (EmployeeID, Name, Position, SSN)

Prescribes = (Physician, Patient, Medication, Date, Appointment, Dose)

Procedures = (Code, Name, Cost)

Room = (RoomNumber, RoomType, BlockFloor, BlockCode, Unavailable)

Stay = (StayID, Patient, Room, StayStart, StayEnd)

Trained_in = (Physician, Treatment, CertificationDate, CertificationExpires)

Undergoes = (Patient, Procedures, Stay, DateUndergoes, Physician, AssistingNurse)

III. Soal

(**Note:** Pastikan bahwa pada komputer yang digunakan, telah terdapat sebuah database bernama hospital. Jika belum, buatlah sebuah database bernama **hospital** dan import hospital.sql ke dalam database tersebut!)

1. Buatlah sebuah database bernama **rumahsakit** yang merupakan duplikat dari database **hospital**.
2. Query berikut memakan waktu yang cukup lama saat dieksekusi

```
SELECT m.name, (SELECT COUNT(*)
  FROM prescribes p
 WHERE p.medication = m.code AND p.date > '2018-02-27'
      AND p.date < '2018-03-05') AS count_medication
FROM medication m;
```

- a. Terapkan index composite pada basis data untuk mengoptimasi waktu query tersebut. Sertakan penjelasan kolom apa saja yang di-index.
- b. Analisa dan jelaskan hasil perbandingan waktu eksekusi query dengan dan tanpa index

Tampilkan screenshot perbandingan eksekusi query

Gunakan command EXPLAIN ANALYZE untuk menganalisis waktu eksekusi query

Jawaban:

Query Pembuatan Index	CREATE INDEX idx_medication ON prescribes (medication); CREATE INDEX idx_date ON prescribes (date);
Penjelasan Index	Kolom yang di-index adalah kolom medication dan date pada tabel prescribes. Hal ini dilakukan karena kedua tabel ini dipakai pada query sebagai pembanding.
SS Sebelum Index	

<pre> rumahsakit=# EXPLAIN ANALYZE rumahsakit=# SELECT m.name, (SELECT COUNT(*) rumahsakit=# FROM prescribes p rumahsakit=# WHERE p.medication = m.code AND p.date > '2018-02-27' rumahsakit=# AND p.date < '2018-03-05') AS count_medication rumahsakit=# FROM medication m; </pre> <p style="text-align: center;">QUERY PLAN</p> <pre> ----- Seq Scan on medication m (cost=0.00..49699.31 rows=5 width=86) (actual time=22.372..93.533 rows=5 loops=1) SubPlan 1 -> Aggregate (cost=9939.64..9939.65 rows=1 width=8) (actual time=18.701..18.701 rows=1 loops=5) -> Seq Scan on prescribes p (cost=0.00..9939.35 rows=115 width=0) (actual time=0.443..18.691 rows=104 loops=5) Filter: ((date > '2018-02-27 00:00:00'::timestamp without time zone) AND (date < '2018-03-05 00:00:00'::timestamp without time zone) AND (medication = m.code)) Rows Removed by Filter: 399802 Planning Time: 0.231 ms Execution Time: 93.583 ms (8 rows) Time: 94,305 ms rumahsakit=# </pre>	
SS Sesudah Index	
<pre> rumahsakit=# EXPLAIN ANALYZE rumahsakit=# SELECT m.name, (SELECT COUNT(*) rumahsakit=# FROM prescribes p rumahsakit=# WHERE p.medication = m.code AND p.date > '2018-02-27' rumahsakit=# AND p.date < '2018-03-05') AS count_medication rumahsakit=# FROM medication m; </pre> <p style="text-align: center;">QUERY PLAN</p> <pre> ----- Seq Scan on medication m (cost=0.00..7275.35 rows=5 width=86) (actual time=0.614..2.228 rows=5 loops=1) SubPlan 1 -> Aggregate (cost=1454.85..1454.86 rows=1 width=8) (actual time=0.443..0.443 rows=1 loops=5) -> Bitmap Heap Scan on prescribes p (cost=14.18..1454.56 rows=115 width=0) (actual time=0.118..0.434 rows=104 loops=5) Recheck Cond: ((date > '2018-02-27 00:00:00'::timestamp without time zone) AND (date < '2018-03-05 00:00:00'::timestamp without time zone)) Filter: (medication = m.code) Rows Removed by Filter: 416 Heap Blocks: exact=2395 -> Bitmap Index Scan on idx_date (cost=0.00..14.15 rows=573 width=0) (actual time=0.062..0.062 rows=520 loops=5) Index Cond: ((date > '2018-02-27 00:00:00'::timestamp without time zone) AND (date < '2018-03-05 00:00:00'::timestamp without time zone)) Planning Time: 0.245 ms Execution Time: 2.254 ms (12 rows) Time: 2,892 ms rumahsakit=# </pre>	
Perbandingan	<p>Terlihat bahwa query pada kolom yang telah diberikan index jauh lebih cepat dibandingkan kolom tanpa index. Hal ini terjadi karena sebelum diberikan index, dilakukan full scan table.</p>

- Pada bagian ini Anda akan melakukan eksekusi tiga jenis query yang memiliki hasil yang sama dan menganalisis kinerja untuk masing-masing query. Analisis kinerja DBMS terhadap ketiga query menggunakan fitur yang tersedia pada PostgreSQL. Berikan penjelasan mengenai hal-hal berikut.

- Proses eksekusi masing-masing query (Gunakan query EXPLAIN ANALYZE)
- Waktu eksekusi masing-masing query
- Analisis perbandingan ketiga query

Query 1

```

SELECT Physician.EmployeeID, Physician.Name, COUNT(Patient.SSN) AS
"Number_of_Patient"

```

```

FROM prescribes
NATURAL JOIN stay
INNER JOIN patient ON patient.ssn = stay.patient
INNER JOIN physician ON physician.employeeID = patient.pcp
GROUP BY Physician.EmployeeID
ORDER BY Physician.EmployeeID ASC;

```

Query 2

```

SELECT Physician.EmployeeID, Physician.Name, pid.number
FROM Physician JOIN (SELECT PCP, COUNT(Patient.SSN) AS number
FROM Prescribes
NATURAL JOIN Stay
INNER JOIN Patient ON Patient.SSN = Stay.Patient GROUP BY PCP) pid
ON (Physician.EmployeeID = pid.PCP)
ORDER BY Physician.EmployeeID ASC;

```

Query 3

```

WITH X AS (SELECT patient.ssn, patient.pcp
            FROM patient
            INNER JOIN prescribes
            ON patient.ssn = prescribes.patient
            NATURAL JOIN stay)

SELECT Physician.EmployeeID, Physician.name, COUNT(X.ssn) AS
"Number_of_Patient"
FROM physician
INNER JOIN X ON physician.employeeId = X.pcp
GROUP BY Physician.EmployeeID
ORDER BY Physician.EmployeeID ASC;

```

Jawaban:

Waktu Eksekusi & Proses Eksekusi Query 1 (SS)

```
labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~/Downloads
Time: 0,727 ms
runahsakit=# EXPLAIN ANALYZE
runahsakit=# SELECT Physician.EmployeeID, Physician.Name, COUNT(Patient.SSN) AS "Number_of_Patient"
runahsakit=# FROM prescribes
runahsakit=# NATURAL JOIN stay
runahsakit=# INNER JOIN patient ON patient.ssn = stay.patient
runahsakit=# INNER JOIN physician ON physician.employeeid = patient.pcp
runahsakit=# GROUP BY Physician.EmployeeID
runahsakit=# ORDER BY Physician.EmployeeID ASC;

QUERY PLAN

-----
Finalize GroupAggregate (cost=680039.36..680052.36 rows=100 width=26) (actual time=4022.549..4022.623 rows=100 loops=1)
  Group Key: physician.employeeid
    -> Gather Merge (cost=680039.36..680050.86 rows=100 width=26) (actual time=4022.544..4022.596 rows=200 loops=1)
      Workers Planned: 1
      Workers Launched: 1
      -> Sort (cost=679039.35..679039.60 rows=100 width=26) (actual time=3994.946..3994.953 rows=100 loops=2)
        Sort Key: physician.employeeid
        Sort Method: quicksort Memory: 32kB
        Worker 0: Sort Method: quicksort Memory: 32kB
        -> Partial HashAggregate (cost=679035.03..679036.03 rows=100 width=26) (actual time=3994.917..3994.928 rows=100 loops=2)
          Group Key: physician.employeeid
          -> Hash Join (cost=4231.71..483072.19 rows=39192569 width=22) (actual time=198.708..1884.636 rows=33328118 loops=2)
            Hash Cond: (prescribes.patient = stay.patient)
            -> Parallel Seq Scan on prescribes (cost=0.00..5293.39 rows=235239 width=4) (actual time=166.714..179.191 rows=199953 loops=2)
            -> Hash (cost=2297.72..2297.72 rows=99999 width=26) (actual time=31.870..31.873 rows=99999 loops=2)
              Buckets: 65536 Batches: 2 Memory Usage: 3528kB
              -> Hash Join (cost=23.75..2297.72 rows=99999 width=26) (actual time=0.123..21.195 rows=99999 loops=2)
                Hash Cond: (patient.pcp = physician.employeeid)
                -> Hash Join (cost=20.50..2020.85 rows=99999 width=12) (actual time=0.090..13.364 rows=99999 loops=2)
                  Hash Cond: (stay.patient = patient.ssn)
                  -> Seq Scan on stay (cost=0.00..1735.99 rows=99999 width=4) (actual time=0.086..4.619 rows=99999 loops=2)
                  -> Hash (cost=13.00..13.00 rows=600 width=8) (actual time=0.080..0.081 rows=600 loops=2)
                    Buckets: 1024 Batches: 1 Memory Usage: 32kB
                    -> Seq Scan on patient (cost=0.00..13.00 rows=600 width=8) (actual time=0.025..0.033 rows=600 loops=2)
```

```
labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~/Downloads

Workers Planned: 1
Workers Launched: 1
-> Sort (cost=679039.35..679039.60 rows=100 width=26) (actual time=3994.946..3994.953 rows=100 loops=2)
  Sort Key: physician.employeeid
  Sort Method: quicksort Memory: 32kB
  Worker 0: Sort Method: quicksort Memory: 32kB
  -> Partial HashAggregate (cost=679035.03..679036.03 rows=100 width=26) (actual time=3994.917..3994.928 rows=100 loops=2)
    Group Key: physician.employeeid
    -> Hash Join (cost=4231.71..483072.19 rows=39192569 width=22) (actual time=198.708..1884.636 rows=33328118 loops=2)
      Hash Cond: (prescribes.patient = stay.patient)
      -> Parallel Seq Scan on prescribes (cost=0.00..5293.39 rows=235239 width=4) (actual time=166.714..179.191 rows=199953 loops=2)
      -> Hash (cost=2297.72..2297.72 rows=99999 width=26) (actual time=31.870..31.873 rows=99999 loops=2)
        Buckets: 65536 Batches: 2 Memory Usage: 3528kB
        -> Hash Join (cost=23.75..2297.72 rows=99999 width=26) (actual time=0.123..21.195 rows=99999 loops=2)
          Hash Cond: (patient.pcp = physician.employeeid)
          -> Hash Join (cost=20.50..2020.85 rows=99999 width=12) (actual time=0.090..13.364 rows=99999 loops=2)
            Hash Cond: (stay.patient = patient.ssn)
            -> Seq Scan on stay (cost=0.00..1735.99 rows=99999 width=4) (actual time=0.086..4.619 rows=99999 loops=2)
            -> Hash (cost=13.00..13.00 rows=600 width=8) (actual time=0.080..0.081 rows=600 loops=2)
              Buckets: 1024 Batches: 1 Memory Usage: 32kB
              -> Seq Scan on patient (cost=0.00..13.00 rows=600 width=8) (actual time=0.025..0.033 rows=600 loops=2)
              -> Hash (cost=2.00..2.00 rows=100 width=18) (actual time=0.029..0.030 rows=100 loops=2)
                Buckets: 1024 Batches: 1 Memory Usage: 13kB
                -> Seq Scan on physician (cost=0.00..2.00 rows=100 width=18) (actual time=0.015..0.028 rows=100 loops=2)

Planning Time: 1.485 ms
JIT:
  Functions: 67
  Options: Inlining true, Optimization true, Expressions true, Deforming true
  Timing: Generation 5.997 ms, Inlining 31.904 ms, Optimization 185.529 ms, Emission 115.650 ms, Total 339.081 ms
Execution Time: 4027.157 ms
(33 rows)

Time: 4029,343 ms (00:04,029)
runahsakit=#
```

Waktu Eksekusi & Proses Eksekusi Query 2 (SS)

```

labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~/Downloads

runhaskit=# EXPLAIN ANALYZE
runhaskit=# SELECT physician.EmployeeID, Physician Name, pid.number
runhaskit=# FROM Physician JOIN (SELECT PCP, COUNT(Patient.SSN) AS number
runhaskit=# FROM Prescribes
runhaskit=# NATURAL JOIN Stay
runhaskit=# INNER JOIN Patient ON Patient.SSN = Stay.Patient GROUP BY PCP) pid
runhaskit=# ON (Physician.EmployeeID = pid.PCP)
runhaskit=# ORDER BY Physician.EmployeeID ASC;

QUERY PLAN

-----
Merge Join  (cost=679377.81..679393.56 rows=100 width=26) (actual time=4126.684..4126.777 rows=100 loops=1)
  Merge Cond: (physician.employeeid = patient.pcp)
    -> Sort  (cost=5.32..5.57 rows=100 width=18) (actual time=153.262..153.266 rows=100 loops=1)
      Sort Key: physician.employeeid
      Sort Method: quicksort Memory: 32kB
      -> Seq Scan on physician  (cost=0.00..2.00 rows=100 width=18) (actual time=153.246..153.252 rows=100 loops=1)
    -> Finalize GroupAggregate  (cost=679372.49..679385.49 rows=100 width=12) (actual time=3973.415..3973.492 rows=100 loops=1)
      Group Key: patient.pcp
      -> Gather Merge  (cost=679372.49..679383.99 rows=100 width=12) (actual time=3973.410..3973.455 rows=200 loops=1)
        Workers Planned: 1
        Workers Launched: 1
        -> Sort  (cost=678372.48..678372.73 rows=100 width=12) (actual time=3859.334..3859.340 rows=100 loops=2)
          Sort Key: patient.pcp
          Sort Method: quicksort Memory: 29kB
          Worker 0: Sort Method: quicksort Memory: 29kB
          -> Partial HashAggregate  (cost=678368.16..678369.16 rows=100 width=12) (actual time=3859.299..3859.316 rows=100 loops=2)
            Group Key: patient.pcp
            -> Hash Join  (cost=6759.83..482405.31 rows=39192569 width=0) (actual time=90.032..1782.064 rows=33328118 loops=2)
              Hash Cond: (prescribes.patient = stay.patient)
              -> Parallel Seq Scan on prescribes  (cost=0.00..5293.39 rows=235239 width=4) (actual time=67.451..759.798 rows=199953 loops=2)
                -> Hash  (cost=2020.85..2020.85 rows=99999 width=12) (actual time=22.398..22.399 rows=99999 loops=2)
                  Buckets: 131072 Batches: 2 Memory Usage: 3218KB
                  -> Hash Join  (cost=20.50..2020.85 rows=99999 width=12) (actual time=0.101..13.366 rows=99999 loops=2)
                    Hash Cond: (stay.patient = patient.ssn)

```

```

labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~/Downloads
Sort Method: quicksort Memory: 32kB
-> Seq Scan on physician (cost=0.00..2.00 rows=100 width=18) (actual time=153.246..153.252 rows=100 loops=1)
-> Finalize GroupAggregate (cost=679372.49..679385.49 rows=100 width=12) (actual time=3973.415..3973.492 rows=100 loops=1)
  Group Key: patient.pcp
  -> Gather Merge (cost=679372.49..679383.99 rows=100 width=12) (actual time=3973.410..3973.455 rows=200 loops=1)
    Workers Planned: 1
    Workers Launched: 1
    -> Sort (cost=678372.48..678372.73 rows=100 width=12) (actual time=3859.334..3859.340 rows=100 loops=2)
      Sort Key: patient.pcp
      Sort Method: quicksort Memory: 29kB
      Worker 0: Sort Method: quicksort Memory: 29kB
      -> Partial HashAggregate (cost=678368.16..678369.16 rows=100 width=12) (actual time=3859.299..3859.316 rows=100 loops=2)
        Group Key: patient.pcp
        -> Hash Join (cost=3759.83..482405.31 rows=39192569 width=8) (actual time=90.032..1782.064 rows=33328118 loops=2)
          Hash Cond: (prescribes.patient = stay.patient)
          -> Parallel Seq Scan on prescribes (cost=0.00..5293.39 rows=235239 width=4) (actual time=67.451..79.798 rows=199953 loops=2)
          -> Hash (cost=2020.85..2020.85 rows=99999 width=12) (actual time=22.398..22.399 rows=99999 loops=2)
            Buckets: 131072 Batches: 2 Memory Usage: 3218kB
            -> Hash Join (cost=20.50..2020.85 rows=99999 width=12) (actual time=0.101..13.366 rows=99999 loops=2)
              Hash Cond: (stay.patient = patient.ssn)
              -> Seq Scan on stay (cost=0.00..1755.99 rows=99999 width=4) (actual time=0.014..4.840 rows=99999 loops=2)
              -> Hash (cost=13.00..13.00 rows=600 width=8) (actual time=0.083..0.084 rows=600 loops=2)
                Buckets: 1024 Batches: 1 Memory Usage: 32kB
                -> Seq Scan on patient (cost=0.00..13.00 rows=600 width=8) (actual time=0.007..0.045 rows=600 loops=2)
Planning Time: 1.064 ms
JIT:
  Functions: 58
Options: Inlining true, Optimization true, Expressions true, Deforming true
Timing: Generation 5.883 ms, Inlining 32.036 ms, Optimization 159.047 ms, Emission 96.090 ms, Total 293.055 ms
Execution Time: 4131.627 ms
(34 rows)

Time: 4133.376 ms (00:04.133)
runhasakt=#

```


Waktu Eksekusi & Proses Eksekusi Query 3 (SS)

```
labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~/Downloads
Time: 4133,376 ms (00:04,133)
runahsakit=# EXPLAIN ANALYZE
runahsakit=# WITH X AS (SELECT patient.ssn, patient.pcp
runahsakit=# FROM patient
runahsakit=# INNER JOIN prescribes
runahsakit=# ON patient.ssn = prescribes.patient
runahsakit=# NATURAL JOIN stay)
runahsakit=#
runahsakit=# SELECT Physician.EmployeeID, Physician.name, COUNT(X.ssn) AS "Number_of_Patient"
runahsakit=# FROM physician
runahsakit=# INNER JOIN X ON physician.employeeid = X.pcp
runahsakit=# GROUP BY Physician.EmployeeID
runahsakit=# ORDER BY Physician.EmployeeID ASC;

QUERY PLAN

-----
Finalize GroupAggregate (cost=650837.13..650850.13 rows=100 width=26) (actual time=4007.575..4007.660 rows=100 loops=1)
  Group Key: physician.employeeid
    -> Gather Merge (cost=650848.63 rows=100 width=26) (actual time=4007.570..4007.633 rows=200 loops=1)
      Workers Planned: 1
      Workers Launched: 1
      -> Sort (cost=649837.12..649837.37 rows=100 width=26) (actual time=3987.671..3987.677 rows=100 loops=2)
        Sort Key: physician.employeeid
        Sort Method: quicksort Memory: 32kB
        Worker 0: Sort Method: quicksort Memory: 32kB
        -> Partial HashAggregate (cost=649832.80..649833.80 rows=100 width=26) (actual time=3987.631..3987.650 rows=100 loops=2)
          Group Key: physician.employeeid
          -> Hash Join (cost=4231.71..453882.41 rows=39206078 width=22) (actual time=207.316..1894.529 rows=33328118 loops=2)
            Hash Cond: (prescribes.patient = patient.ssn)
            -> Parallel Seq Scan on prescribes (cost=0.00..5293.39 rows=235239 width=4) (actual time=174.350..187.108 rows=19953 loops=2)
            -> Hash (cost=2297.72..2297.72 rows=99999 width=26) (actual time=32.841..32.843 rows=99999 loops=2)
              Buckets: 65536 Batches: 2 Memory Usage: 3620kB
              -> Hash Join (cost=23.75..2297.72 rows=99999 width=26) (actual time=0.123..21.927 rows=99999 loops=2)
                Hash Cond: (patient.pcp = physician.employeeid)
                -> Hash Join (cost=20.50..2020.85 rows=99999 width=12) (actual time=0.090..13.665 rows=99999 loops=2)
                  Hash Cond: (stay.patient = patient.ssn)
                  -> Seq Scan on stay (cost=0.00..1735.99 rows=99999 width=4) (actual time=0.006..4.813 rows=99999 loops=2)
                  -> Hash (cost=13.00..13.00 rows=600 width=8) (actual time=0.081..0.082 rows=600 loops=2)
                    Buckets: 1024 Batches: 1 Memory Usage: 32kB
                    -> Seq Scan on patient (cost=0.00..13.00 rows=600 width=8) (actual time=0.005..0.043 rows=600 loops=2)
                    -> Hash (cost=2.00..2.00 rows=100 width=18) (actual time=0.029..0.029 rows=100 loops=2)
                      Buckets: 1024 Batches: 1 Memory Usage: 13kB
                      -> Seq Scan on physician (cost=0.00..2.00 rows=100 width=10) (actual time=0.015..0.020 rows=100 loops=2)
                      Planning Time: 1.442 ms
                      JIT:
                        Functions: 67
                        Options: Inlining true, Optimization true, Expressions true, Deforming true
                        Timing: Generation 5.981 ms, Inlining 32.549 ms, Optimization 194.274 ms, Emission 121.450 ms, Total 354.254 ms
                        Execution Time: 4012.282 ms
                        (33 rows)
Time: 4014,459 ms (00:04,014)
runahsakit=#
```

Analisis Perbandingan Query

Dari ketiga query tersebut, terlihat bahwa kecepatan query 3 < query 2 < query 1 (artinya query 3 lebih cepat daripada query 1 dan 2). Hal ini dapat ditinjau dari jumlah row tiap tabel. Diketahui jumlah row pada tabel prescribes = physician = 399906, tabel stay = 99999, dan tabel patient = 600.

Query 1
Awalnya dilakukan natural join antara prescribes dan stay. Jumlah I/O cost adalah 399906 + 99999. Jumlah record yang terbentuk adalah 99999. Lalu, dilakukan join terhadap patient sehingga jumlah I/O cost sekitar 399906 + 99999 + 99999 + 600. Jumlah record yang terbentuk adalah 600. Terakhir, dilakukan join terhadap physician sehingga jumlah total I/O cost adalah 399906 + 99999 + 99999 + 600 + 600 + 399906.

Query 2
Awalnya dilakukan natural join antara prescribes dan stay. Jumlah I/O cost adalah 399906 + 99999. Jumlah record yang terbentuk

adalah 99999. Lalu, dilakukan join terhadap patient sehingga jumlah I/O cost sekitar $399906 + 99999 + 99999 + 600$. Jumlah record yang terbentuk adalah 600. Setelah itu, di-select pcp dan count dari patient. Artinya, seluruh rows tersebut ditambahkan kolom count yang berisi jumlah pasien yang berobat. Terakhir, dilakukan join terhadap physician sehingga jumlah total I/O cost adalah $399906 + 99999 + 99999 + 600 + 600 + 399906$. Cost seharusnya lebih mahal daripada query 1 akibat scanning count yang memuat jumlah rows yang lebih besar dibandingkan query 1.

Query 3

Awalnya dilakukan join antara patient dan prescribes. Jumlah I/O cost adalah $600 + 399906$. Jumlah record yang terbentuk adalah 600. Lalu, dilakukan join terhadap stay sehingga jumlah I/O cost menjadi $600 + 399906 + 600 + 99999$. Jumlah record yang terbentuk adalah 600. Terakhir, dilakukan join terhadap physician sehingga jumlah I/O cost menjadi $600 + 399906 + 600 + 99999 + 600 + 399906$. Total I/O cost ini adalah yang paling murah dibandingkan query 1 dan 2.

4. Analisis dan eksekusi query berikut pada DBMS.

```
WITH X AS
(SELECT a.physician, a.department, d.name
FROM affiliated_with a, department d
WHERE a.department = d.departmentid),

Y AS
(SELECT physician.employeeid, physician.name, COUNT(patient.SSN)AS
total_patient
FROM physician, patient
WHERE physician.employeeid = patient.pcp
GROUP BY (physician.employeeid)
),

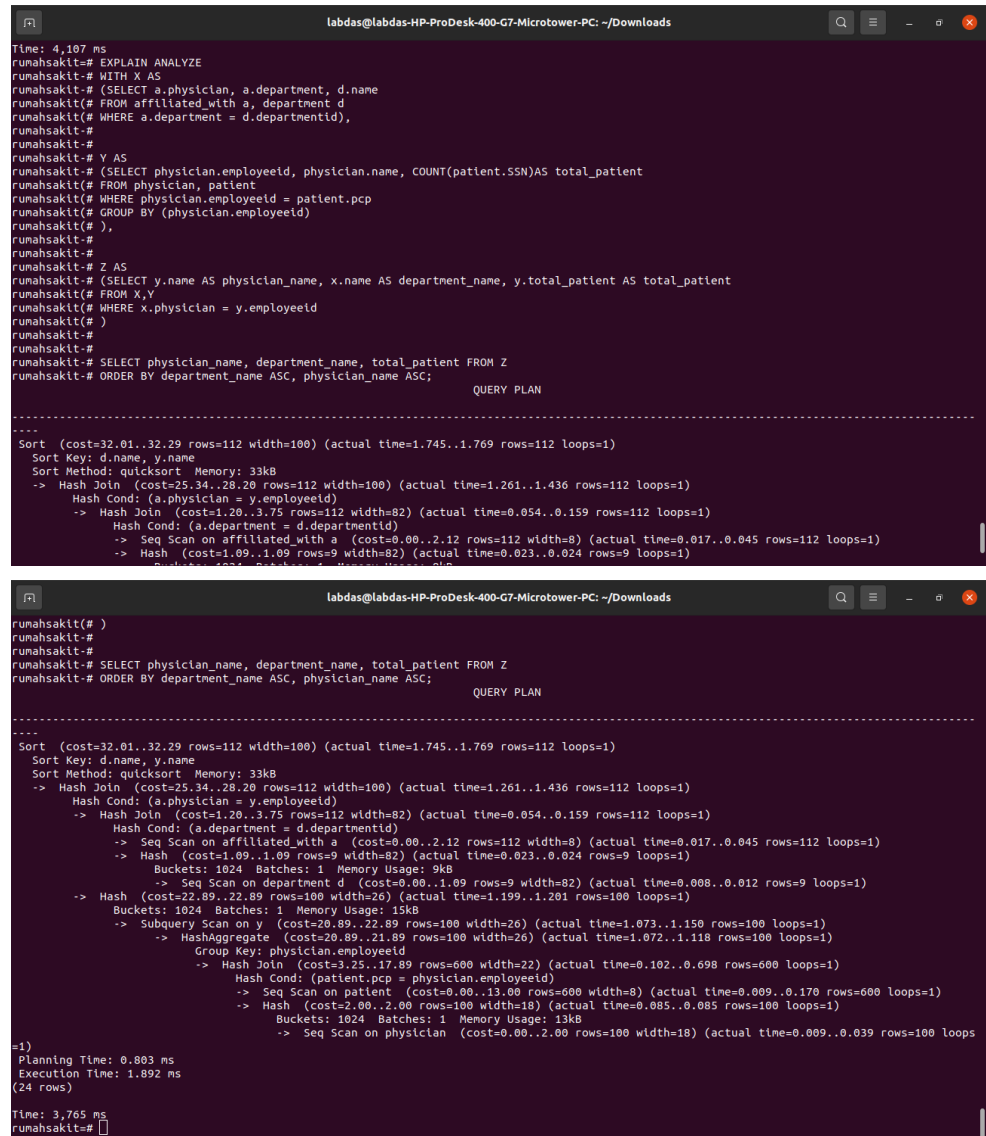
Z AS
(SELECT y.name AS physician_name, x.name AS department_name, y.total_patient AS
total_patient
FROM X,Y
WHERE x.physician = y.employeeid
)
```

```
SELECT physician_name, department_name, total_patient FROM Z
ORDER BY department_name ASC, physician_name ASC;
```

Melalui analisa Anda, apakah terdapat query yang memberikan output yang sama dengan query di atas tetapi memiliki waktu eksekusi yang lebih cepat? Jika ada, buktikan dengan memberikan satu query yang menurut anda paling efisien tanpa mengubah semantik dari query yang ada sebelumnya. Sertakan juga analisis mengapa query tersebut bisa lebih efisien.

Jawaban:

**Waktu
Eksekusi &
Proses
Eksekusi Query
Soal
(screenshot)**



```
labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~/Downloads
Time: 4,107 ms
runahsakti=# EXPLAIN ANALYZE
runahsakti=# WITH X AS
runahsakti=# (SELECT a.physician, a.department, d.name
runahsakti=# FROM affiliated_with a, department d
runahsakti=# WHERE a.department = d.departmentid),
runahsakti=#
runahsakti=# Y AS
runahsakti=# (SELECT physician.employeeid, physician.name, COUNT(patient.SSN)AS total_patient
runahsakti=# FROM physician, patient
runahsakti=# WHERE physician.employeeid = patient.pcp
runahsakti=# GROUP BY (physician.employeeid)
runahsakti=# ),
runahsakti=#
runahsakti=# Z AS
runahsakti=# (SELECT y.name AS physician_name, x.name AS department_name, y.total_patient AS total_patient
runahsakti=# FROM X,y
runahsakti=# WHERE x.physician = y.employeeid
runahsakti=# )
runahsakti=#
runahsakti=# SELECT physician_name, department_name, total_patient FROM Z
runahsakti=# ORDER BY department_name ASC, physician_name ASC;
QUERY PLAN

-----
Sort (cost=32.01..32.29 rows=112 width=100) (actual time=1.745..1.769 rows=112 loops=1)
  Sort Key: d.name, y.name
  Sort Method: quicksort  Memory: 33kB
  -> Hash Join (cost=25.34..28.20 rows=112 width=100) (actual time=1.261..1.436 rows=112 loops=1)
    Hash Cond: (a.physician = y.employeeid)
    -> Hash Join (cost=1.20..3.75 rows=112 width=82) (actual time=0.054..0.159 rows=112 loops=1)
      Hash Cond: (a.department = d.departmentid)
      -> Seq Scan on affiliated_with a (cost=0.00..2.12 rows=112 width=8) (actual time=0.017..0.045 rows=112 loops=1)
      -> Hash (cost=1.09..1.09 rows=9 width=82) (actual time=0.023..0.024 rows=9 loops=1)
    -> Seq Scan on patient p (cost=0.00..1.00 rows=100 width=16) (actual time=0.000..0.012 rows=100 loops=1)

Time: 3,765 ms
runahsakti=#
```

```
labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~/Downloads
runahsakti=#
runahsakti=#
runahsakti=# SELECT physician_name, department_name, total_patient FROM Z
runahsakti=# ORDER BY department_name ASC, physician_name ASC;
QUERY PLAN

-----
Sort (cost=32.01..32.29 rows=112 width=100) (actual time=1.745..1.769 rows=112 loops=1)
  Sort Key: d.name, y.name
  Sort Method: quicksort  Memory: 33kB
  -> Hash Join (cost=25.34..28.20 rows=112 width=100) (actual time=1.261..1.436 rows=112 loops=1)
    Hash Cond: (a.physician = y.employeeid)
    -> Hash Join (cost=1.20..3.75 rows=112 width=82) (actual time=0.054..0.159 rows=112 loops=1)
      Hash Cond: (a.department = d.departmentid)
      -> Seq Scan on affiliated_with a (cost=0.00..2.12 rows=112 width=8) (actual time=0.017..0.045 rows=112 loops=1)
      -> Hash (cost=1.09..1.09 rows=9 width=82) (actual time=0.023..0.024 rows=9 loops=1)
    -> Seq Scan on patient p (cost=0.00..1.00 rows=100 width=16) (actual time=0.000..0.012 rows=100 loops=1)
  -> Hash (cost=22.89..22.89 rows=100 width=26) (actual time=1.199..1.201 rows=100 loops=1)
    Buckets: 1024 Batches: 1 Memory Usage: 15kB
    -> Subquery Scan on y (cost=20.89..22.89 rows=100 width=26) (actual time=1.073..1.150 rows=100 loops=1)
      -> HashAggregate (cost=20.89..21.89 rows=100 width=26) (actual time=1.072..1.118 rows=100 loops=1)
        Group Key: physician.employeeid
        -> Hash Join (cost=3.25..17.89 rows=600 width=22) (actual time=0.102..0.698 rows=600 loops=1)
          Hash Cond: (patient.pcp = physician.employeeid)
          -> Seq Scan on patient (cost=0.00..13.00 rows=600 width=8) (actual time=0.009..0.170 rows=600 loops=1)
          -> Hash (cost=2.00..2.00 rows=100 width=18) (actual time=0.085..0.085 rows=100 loops=1)
            Buckets: 1024 Batches: 1 Memory Usage: 13kB
            -> Seq Scan on physician (cost=0.00..2.00 rows=100 width=18) (actual time=0.009..0.039 rows=100 loops=1)

Planning Time: 0.803 ms
Execution Time: 1.892 ms
(24 rows)

Time: 3,765 ms
runahsakti=#
```

Usulan Query

WITH X AS

```
(SELECT a.physician, a.department, d.name
FROM affiliated_with a JOIN department d
ON a.department = d.departmentid),

Y AS
(SELECT physician.employeeid, physician.name, COUNT(patient.SSN)AS
total_patient
FROM physician JOIN patient
ON physician.employeeid = patient.pcp
GROUP BY (physician.employeeid)
),

Z AS
(SELECT y.name AS physician_name, x.name AS department_name,
y.total_patient AS total_patient
FROM X JOIN Y
ON x.physician = y.employeeid
)

SELECT physician_name, department_name, total_patient FROM Z
ORDER BY department_name ASC, physician_name ASC;
```

Waktu
Eksekusi &
Proses
Eksekusi Query
Usulan (SS)

```
labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~/Downloads

runahsakit=# WITH X AS
runahsakit=# (SELECT a.physician, a.department, d.name
runahsakit=# FROM affiliated_with a JOIN department d
runahsakit=# ON a.department = d.departmentid),
runahsakit=#
runahsakit=#
runahsakit=# Y AS
runahsakit=# (SELECT physician.employeeid, physician.name, COUNT(patient.SSN)AS total_patient
runahsakit=# FROM physician JOIN patient
runahsakit=# ON physician.employeeid = patient.pcp
runahsakit=# GROUP BY (physician.employeeid)
runahsakit=# ),
runahsakit=#
runahsakit=#
runahsakit=# Z AS
runahsakit=# (SELECT y.name AS physician_name, x.name AS department_name, y.total_patient AS total_patient
runahsakit=# FROM X JOIN Y
runahsakit=# ON x.physician = y.employeeid
runahsakit=# )
runahsakit=#
runahsakit=# SELECT physician_name, department_name, total_patient FROM Z
runahsakit=# ORDER BY department_name ASC, physician_name ASC;
physician_name | department_name | total_patient
-----
Dallas Blevin | Accident and Emergency | 9
Des Sweedy | Accident and Emergency | 7
Dore Biesterfeld | Accident and Emergency | 4
Elliot Reid | Accident and Emergency | 4
Ethelda Gerrelts | Accident and Emergency | 9
Gert Swaden | Accident and Emergency | 4
Gisele McClinton | Accident and Emergency | 6
Hesther Eyan | Accident and Emergency | 1
John Dorian | Accident and Emergency | 4
John Wen | Accident and Emergency | 9
Joni Hunnerston | Accident and Emergency | 7
Judy Copeland | Accident and Emergency | 8
Lynnette Carthew | Accident and Emergency | 4
(112 rows)

Time: 3,170 ms
runahsakit=# EXPLAIN ANALYZ
runahsakit=# E
runahsakit=# E
```

	<div data-bbox="451 212 1421 766"><pre>labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~/Downloads Time: 0.442 ms runahsakit=# EXPLAIN ANALYZE runahsakit=# WITH X AS runahsakit=# (SELECT a.physician, a.department, d.name runahsakit=# FROM affiliated_with a JOIN department d runahsakit=# ON a.department = d.departmentid), runahsakit=# runahsakit=# Y AS runahsakit=# (SELECT physician.employeeid, physician.name, COUNT(patient.SSN)AS total_patient runahsakit=# FROM physician JOIN patient runahsakit=# ON physician.employeeid = patient.pcp runahsakit=# GROUP BY (physician.employeeid) runahsakit=#), runahsakit=# runahsakit=# Z AS runahsakit=# (SELECT y.name AS physician_name, x.name AS department_name, y.total_patient AS total_patient runahsakit=# FROM X JOIN Y runahsakit=# ON x.physician = y.employeeid runahsakit=#) runahsakit=# runahsakit=# SELECT physician_name, department_name, total_patient FROM Z runahsakit=# ORDER BY department_name ASC, physician_name ASC; QUERY PLAN ----- Sort (cost=32.01..32.29 rows=112 width=100) (actual time=1.729..1.753 rows=112 loops=1) Sort Key: d.name, y.name Sort Method: quicksort Memory: 33kB -> Hash Join (cost=25.34..28.20 rows=112 width=100) (actual time=1.243..1.419 rows=112 loops=1) Hash Cond: (a.physician = y.employeeid) -> Hash Join (cost=1.20..3.75 rows=112 width=82) (actual time=0.051..0.158 rows=112 loops=1) Hash Cond: (a.department = d.departmentid) -> Seq Scan on affiliated_with a (cost=0.00..2.12 rows=112 width=8) (actual time=0.016..0.044 rows=112 loops=1) -> Hash (cost=1.09..1.09 rows=9 width=82) (actual time=0.022..0.023 rows=9 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9kB -> Seq Scan on department d (cost=0.00..1.09 rows=9 width=82) (actual time=0.008..0.012 rows=9 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 15kB -> Subquery Scan on y (cost=20.89..22.89 rows=100 width=26) (actual time=1.060..1.134 rows=100 loops=1) -> HashAggregate (cost=20.89..21.89 rows=100 width=26) (actual time=1.058..1.105 rows=100 loops=1) Group Key: physician.employeeid -> Hash Join (cost=3.25..17.89 rows=600 width=22) (actual time=0.101..0.694 rows=600 loops=1) Hash Cond: (patient.pcp = physician.employeeid) -> Seq Scan on patient (cost=0.00..13.00 rows=600 width=8) (actual time=0.009..0.167 rows=600 loops=1) -> Hash (cost=2.00..2.00 rows=100 width=18) (actual time=0.083..0.084 rows=100 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 13kB -> Seq Scan on physician (cost=0.00..2.00 rows=100 width=18) (actual time=0.009..0.038 rows=100 loops=1) Planning Time: 0.852 ms Execution Time: 1.876 ms (24 rows) Time: 3,848 ms runahsakit=#</pre></div> <div data-bbox="451 787 1421 1341"><pre>labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~/Downloads runahsakit=#) runahsakit=# runahsakit=# SELECT physician_name, department_name, total_patient FROM Z runahsakit=# ORDER BY department_name ASC, physician_name ASC; QUERY PLAN ----- Sort (cost=32.01..32.29 rows=112 width=100) (actual time=1.729..1.753 rows=112 loops=1) Sort Key: d.name, y.name Sort Method: quicksort Memory: 33kB -> Hash Join (cost=25.34..28.20 rows=112 width=100) (actual time=1.243..1.419 rows=112 loops=1) Hash Cond: (a.physician = y.employeeid) -> Hash Join (cost=1.20..3.75 rows=112 width=82) (actual time=0.051..0.158 rows=112 loops=1) Hash Cond: (a.department = d.departmentid) -> Seq Scan on affiliated_with a (cost=0.00..2.12 rows=112 width=8) (actual time=0.016..0.044 rows=112 loops=1) -> Hash (cost=1.09..1.09 rows=9 width=82) (actual time=0.022..0.023 rows=9 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9kB -> Seq Scan on department d (cost=0.00..1.09 rows=9 width=82) (actual time=0.008..0.012 rows=9 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 15kB -> Subquery Scan on y (cost=20.89..22.89 rows=100 width=26) (actual time=1.060..1.134 rows=100 loops=1) -> HashAggregate (cost=20.89..21.89 rows=100 width=26) (actual time=1.058..1.105 rows=100 loops=1) Group Key: physician.employeeid -> Hash Join (cost=3.25..17.89 rows=600 width=22) (actual time=0.101..0.694 rows=600 loops=1) Hash Cond: (patient.pcp = physician.employeeid) -> Seq Scan on patient (cost=0.00..13.00 rows=600 width=8) (actual time=0.009..0.167 rows=600 loops=1) -> Hash (cost=2.00..2.00 rows=100 width=18) (actual time=0.083..0.084 rows=100 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 13kB -> Seq Scan on physician (cost=0.00..2.00 rows=100 width=18) (actual time=0.009..0.038 rows=100 loops=1) Planning Time: 0.852 ms Execution Time: 1.876 ms (24 rows) Time: 3,848 ms runahsakit=#</pre></div>
<p>Penjelasan Query Baru & Analisis Perbandingan</p>	<p>Pada query baru digunakan JOIN sehingga lebih cepat dibandingkan CROSS JOIN (penggunaan klausa WHERE). Hal ini karena jumlah I/O cost pada JOIN jauh lebih sedikit dibandingkan perkalian atau CROSS JOIN.</p>

Feedback Praktikum

bit.ly/feedback-praktikum