Telekom MMS

EXPERIENCE
BEYOND
DIGITAL

Dresdner Datenbankforum

Kein Datenbankschema überlebt den Kontakt mit der Realität: Hier ist der Plan B!

DESC me;

IT-Architekt bei Telekom MMS

- seit 25 Jahren im Ops Umfeld
- verschiedene OLTP-Umgebungen auf Basis von Web-Technologien
- Shops, Firmenseiten, Info Portale, Callcenter, Aggregatoren, Backendsysteme
- Hauptarbeitsfelder: Automatisierung, Performance, Optimierung
- Open Source Arbeit

SELECT name FROM db.software;

CREATE DATABASE production;

https://gunshowcomic.com/648

SELECT * FROM contents;

Tools

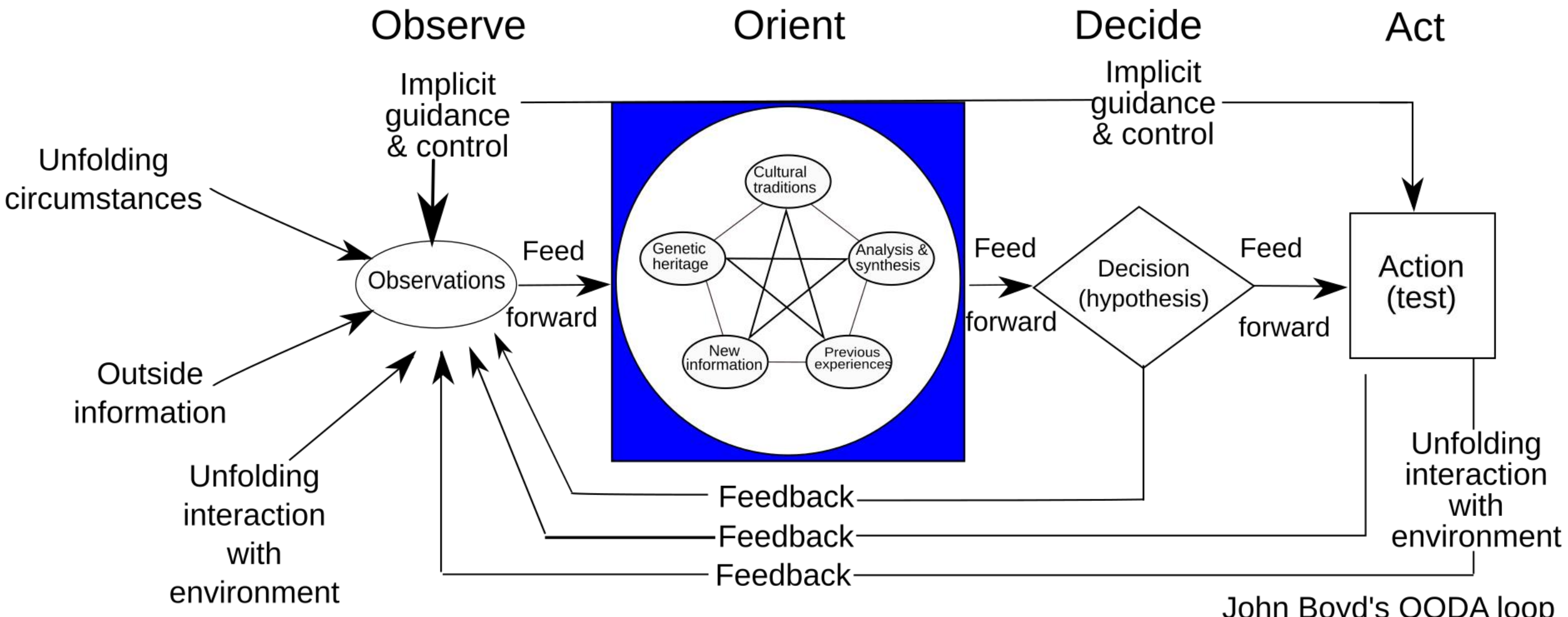
Erfahrungen

Wissen

B, wie

Beobachten

CREATE TRIGGER watch ON env ...



John Boyd's OODA loop

https://en.wikipedia.org/wiki/OODA_loop

INSERT INTO environment VALUES („Linux Tools“)

```
[schurzi@fafnyr ~]$ top
top - 21:27:16 up 301 days,  5:34,  4 users,  load average: 2.42, 2.54, 2.76
Tasks: 222 total, 1 running, 221 sleep, 0 d-sleep, 0 stopped, 0 zombie
%Cpu(s): 36.4 us,  2.1 sy,  0.0 ni, 57.3 id,  1.3 wa,  2.4 hi,  0.3 si,  0.1 st
MiB Mem :   3888.6 total,    378.8 free,   2970.3 used,    799.3 buff/cache
MiB Swap:   2048.0 total,   1653.0 free,    395.0 used.    918.4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
2198172 mysql     20   0 1590136 823800  14092 S 133.5  20.7    16w+1d mariadbd
1553310 netdata   20   0  797368 166436  13468 S   9.3   4.2    7d+14h netdata
1553879 netdata   20   0  103228   8120   3144 S   3.3   0.2     73,31 apps.plugin
 514372 http      20   0  285304  32120  20276 S   2.3   0.8   0:01.70 php-fpm
 513754 http      20   0  283824  18096   8020 S   2.0   0.5   0:02.92 php-fpm
1553875 netdata   20   0 1416104  78524  14876 S   0.7   2.0     18,03 go.d.plugin
    471 root      20   0   16392   4892   4380 S   0.3   0.1     10,53 systemd-logind
 515116 schurzi   20   0   12396   5972   3668 R   0.3   0.1   0:00.12 top
1553881 netdata   20   0  201492   3692   3308 S   0.3   0.1 117:20.07 sd-jrnl.plugin
      1 root      20   0   23172   8296   5092 S   0.0   0.2 149:28.56 systemd
```

```
[schurzi@fafnyr ~]$ vmstat 5
procs -----------memory---------- ---swap-- -----io---- -system-- -------cpu-------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st gu
 3  0 404496 410980 165972 653904    0    0  7216    67 3213    4 11  4 79  6  0  0
 2  0 404496 408232 166004 653912    0    0   134    19 8510 1291 65  6 28  1  0  0
 3  0 404496 409304 166016 653920    0    0     0    79 7945 1171 66  6 27  0  0  0
 3  0 404496 407080 166028 653928    0    0     0    26 4352  957 33  4 63  0  0  0
 3  0 404496 402648 166048 655252    0    0   723    18 8679 1452 56  7 34  2  0  0
 3  0 404496 402040 166052 655260    0    0     9    17 8301 1217 70  6 24  0  0  0
 3  0 404496 401788 166076 655264    0    0     0    40 8210 1077 73  6 20  0  0  0
```

```
[schurzi@fafnyr ~]$ iostat -y -h -x -t 30 1
Linux 6.15.3-arch1-1 (fafnyr)   04/20/2026      _x86_64_        (4 CPU)

04/20/2026 09:37:44 PM
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          64.6%    0.0%   15.5%    3.9%    0.3%   15.7%

     r/s     rkB/s   rrqm/s  %rrqm r_await rareq-sz Device
    0.00      0.0k     0.00   0.0%    0.00      0.0k dm-0
    0.00      0.0k     0.00   0.0%    0.00      0.0k dm-1
 2313.50     36.1M     0.00   0.0%    0.24     16.0k dm-2
    0.00      0.0k     0.00   0.0%    0.00      0.0k dm-3
    4.80    213.1k     0.00   0.0%    4.86     44.4k dm-4
    0.00      0.0k     0.00   0.0%    0.00      0.0k dm-5
 2314.27     36.4M     6.53   0.3%    0.26     16.1k xvda

     w/s     wkB/s   wrqm/s  %wrqm w_await wareq-sz Device
    0.00      0.0k     0.00   0.0%    0.00      0.0k dm-0
    4.33     17.6k     0.00   0.0%    1.99      4.1k dm-1
    8.47     55.6k     0.00   0.0%    1.48      6.6k dm-2
    0.00      0.0k     0.00   0.0%    0.00      0.0k dm-3
    1.67      6.0k     0.00   0.0%    1.70      3.6k dm-4
    0.00      0.0k     0.00   0.0%    0.00      0.0k dm-5
   11.13     79.2k     3.57  24.3%    2.08      7.1k xvda

     d/s     dkB/s   drqm/s  %drqm d_await dareq-sz Device
    0.00      0.0k     0.00   0.0%    0.00      0.0k dm-0
    0.00      0.0k     0.00   0.0%    0.00      0.0k dm-1
    0.00      0.0k     0.00   0.0%    0.00      0.0k dm-2
    0.00      0.0k     0.00   0.0%    0.00      0.0k dm-3
    0.00      0.0k     0.00   0.0%    0.00      0.0k dm-4
    0.00      0.0k     0.00   0.0%    0.00      0.0k dm-5
    0.00      0.0k     0.00   0.0%    0.00      0.0k xvda

     f/s f_await  aqu-sz  %util Device
    0.00    0.00    0.00   0.0% dm-0
    0.00    0.00    0.01   0.1% dm-1
    0.00    0.00    0.56  33.9% dm-2
    0.00    0.00    0.00   0.0% dm-3
    0.00    0.00    0.03   2.3% dm-4
    0.00    0.00    0.00   0.0% dm-5
    2.60    0.67    0.63  36.1% xvda
```

INSERT INTO environment VALUES („metrics“)

INSERT INTO environment VALUES („DB Features“)

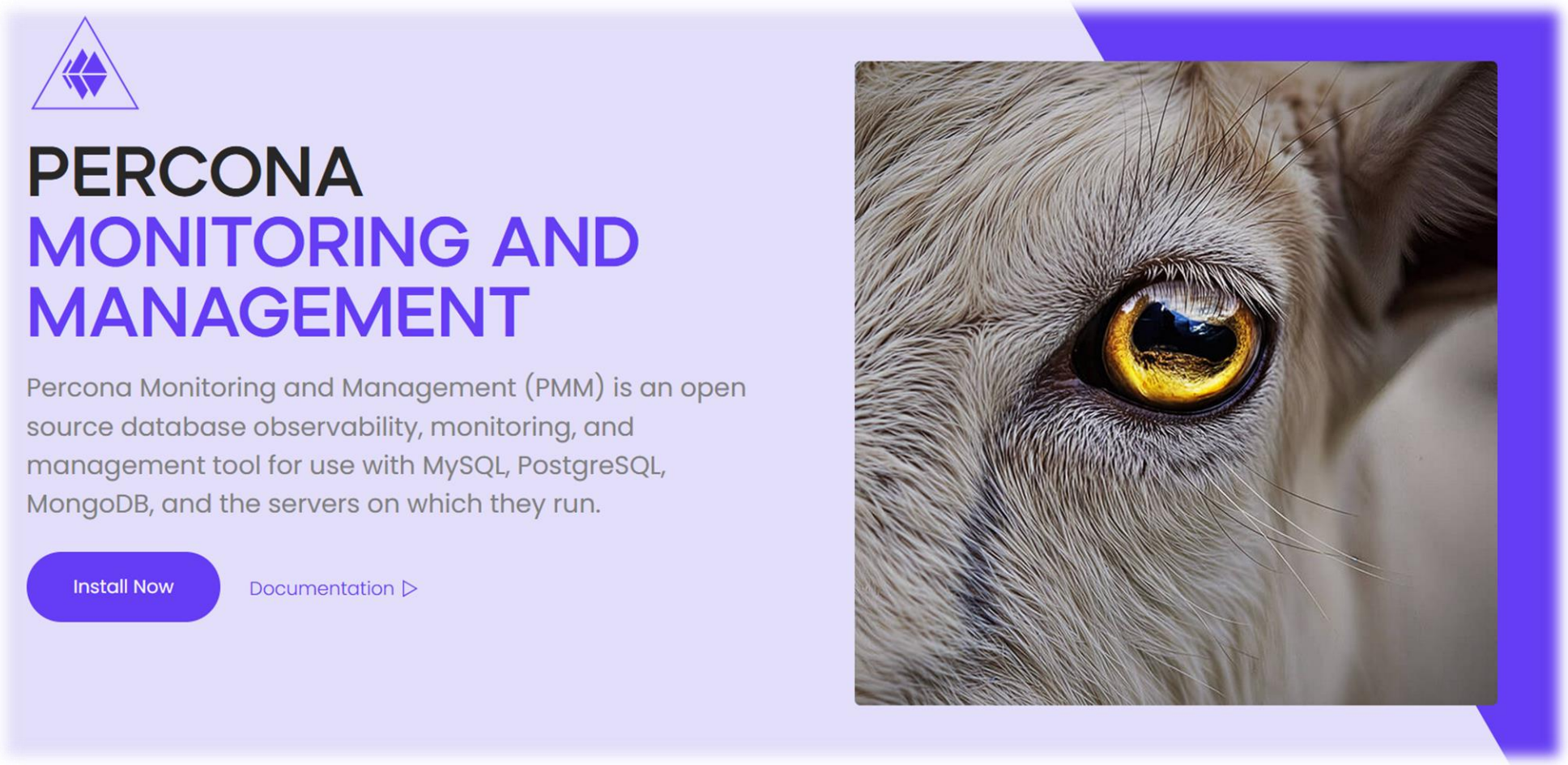
Mit System Metriken kann man erkennen, wo der Engpass ist

Frage: Was konsumiert die Ressourcen?

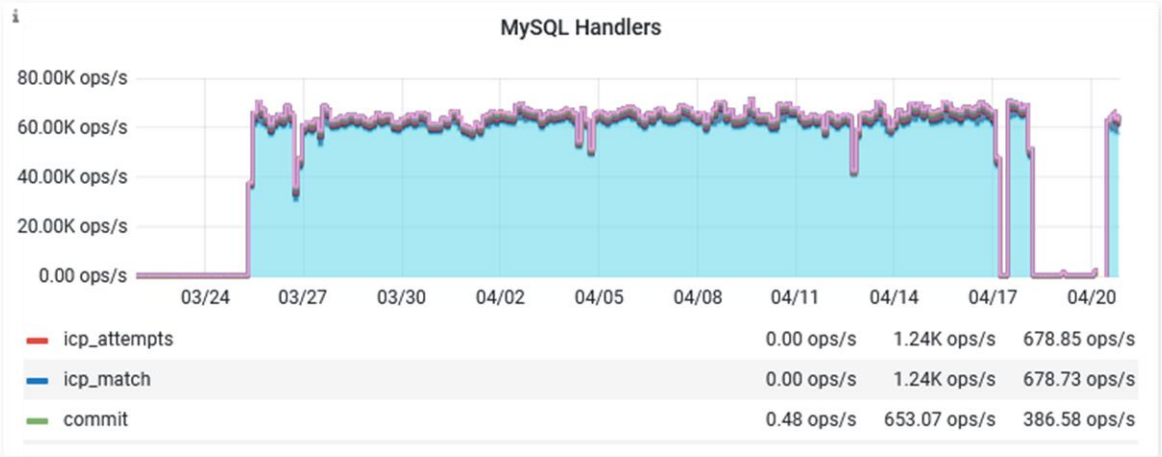
- DBMS interne Metriken
- Ganz besonders Query-Profile
  - PostgreSQL pg_stat_statements Extension
  - ⚠️ MySQL Slow Query Log
  - MongoDB Profiling
- Absolute Arbeitsgrundlage, das MUSS da sein!

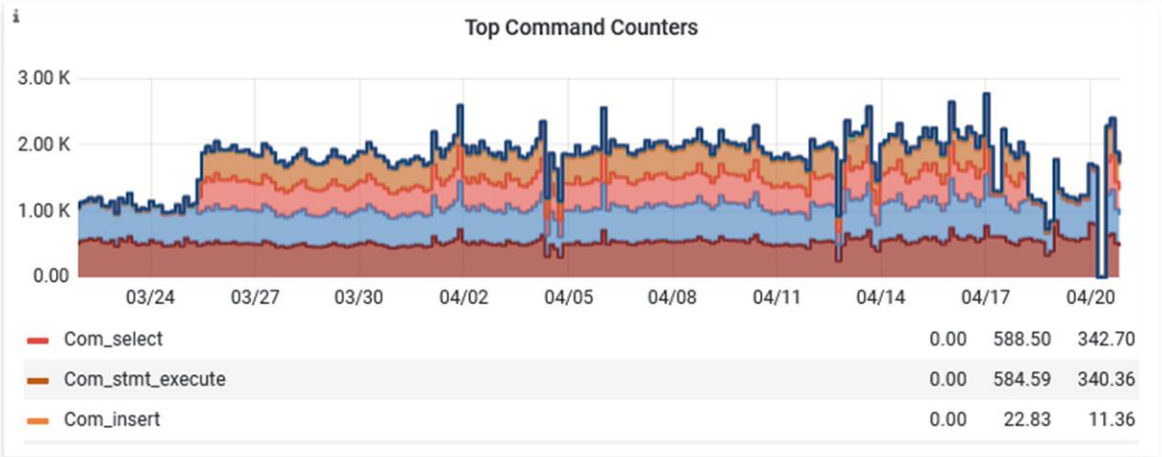
```
employees=# SELECT SUBSTR(query, 1, 15),calls,total_exec_time,mean_exec_time,rows
                 FROM pg_stat_statements ORDER BY total_exec_time DESC;
     substr      | calls |  total_exec_time   |    mean_exec_time    | rows
-----------------+-------+--------------------+----------------------+------
 SELECT d.dept_n |     5 |        1352.686885 |           270.537377 |   25
 SELECT first_na |    29 |           0.856591 |             0.029537 |    0
```

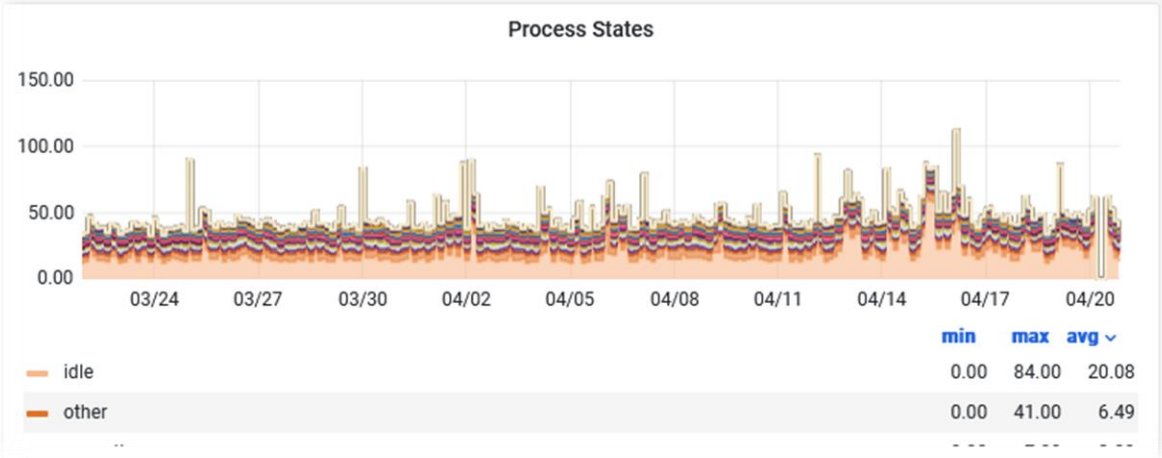
INSERT INTO environment VALUES („metrics“)

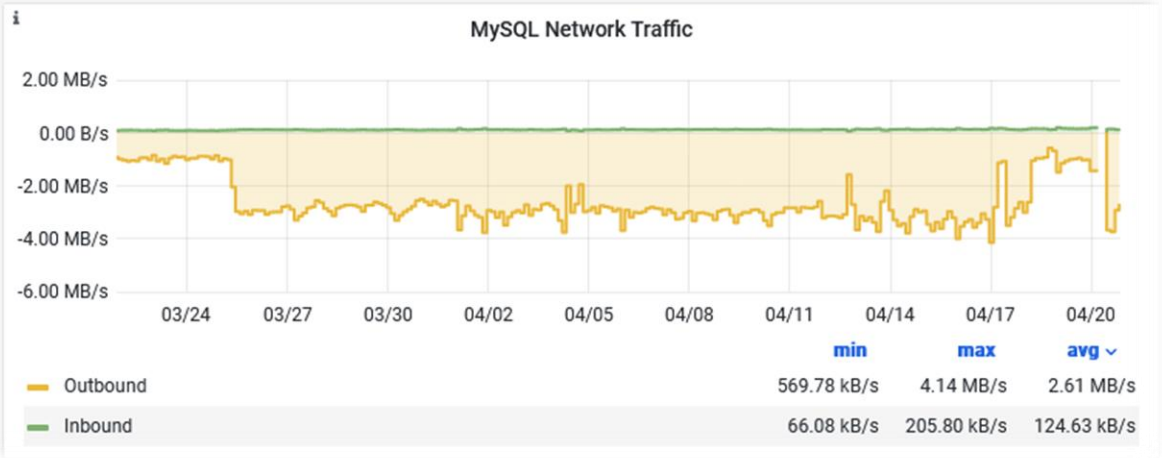


INSERT INTO environment VALUES („metrics“)

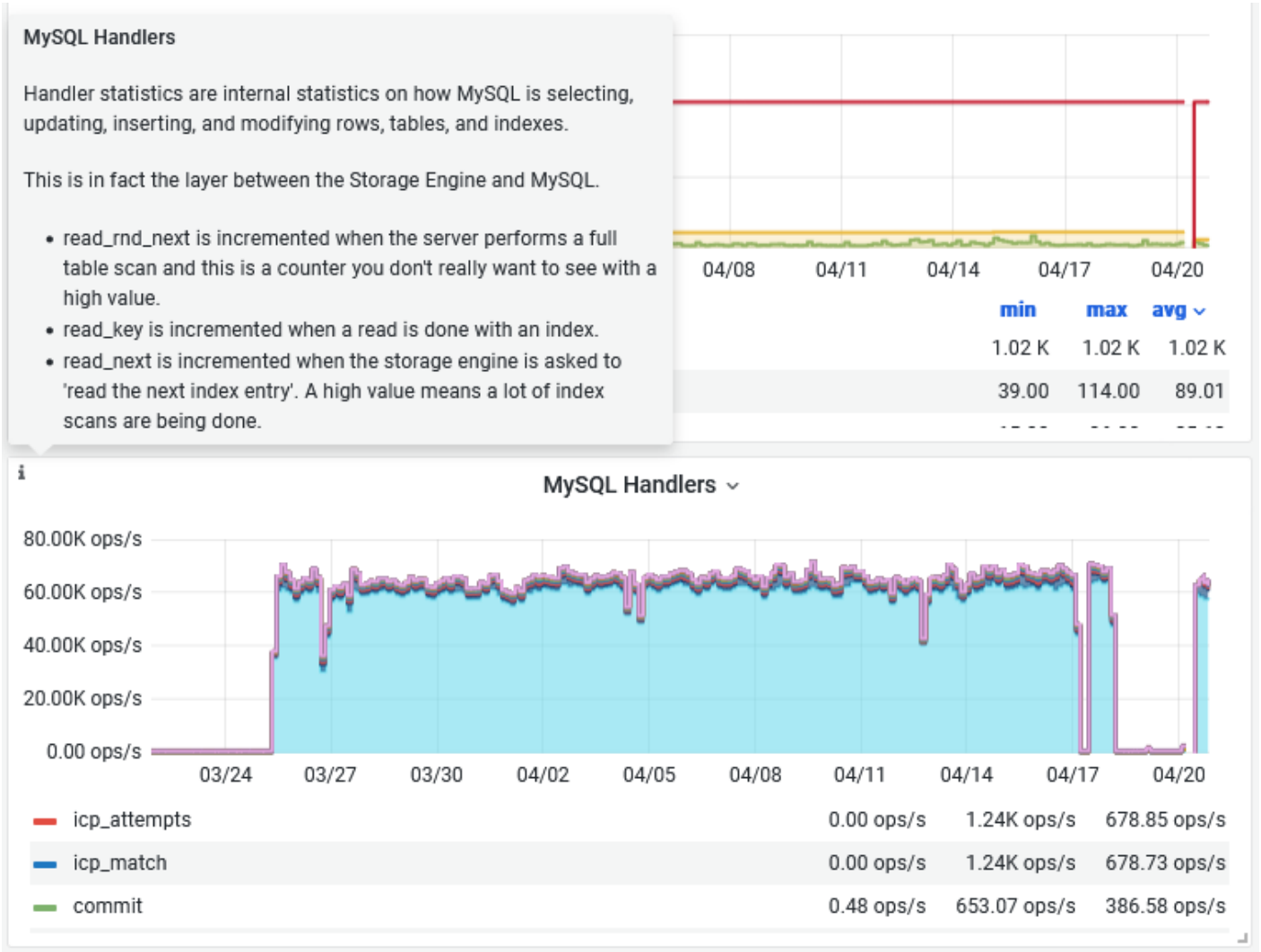




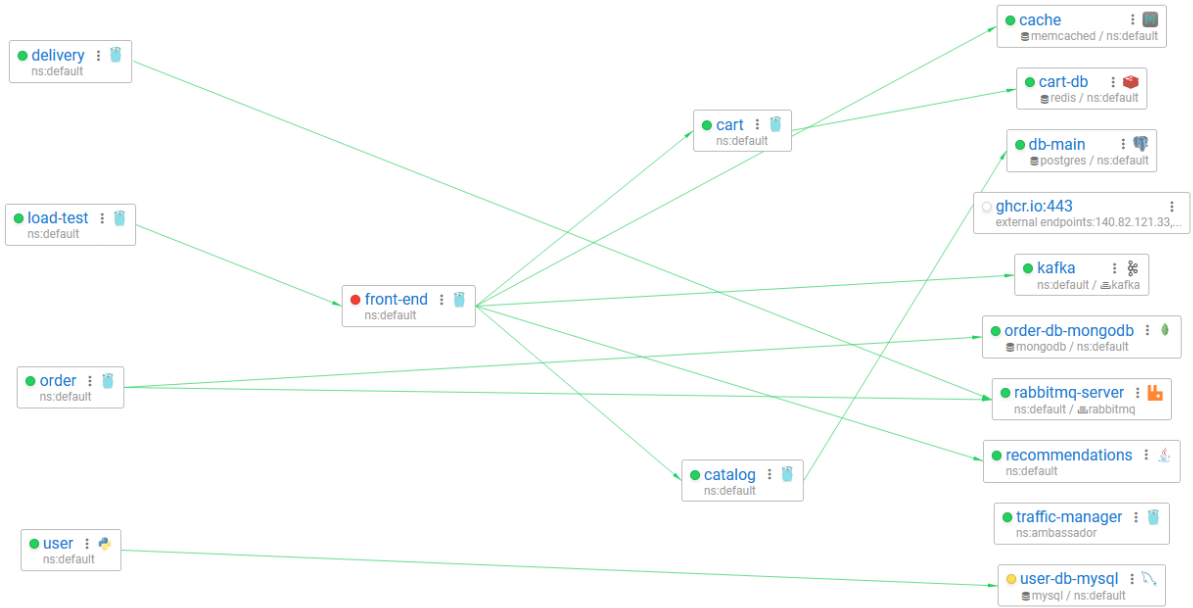


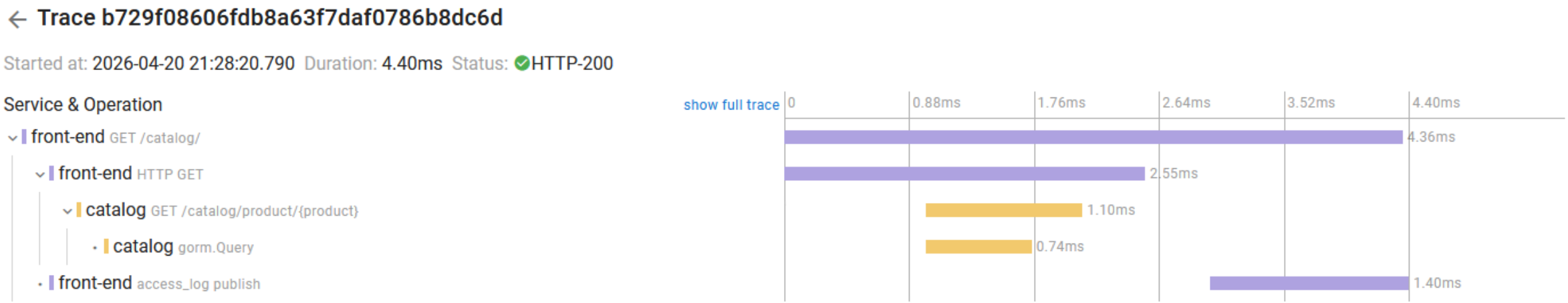


INSERT INTO environment VALUES („metrics“)



INSERT INTO environment VALUES („observability“)





Jetzt wissen wir, was wir wollen
… aber wie?

B, wie

B-Tree

CREATE INDEX ...

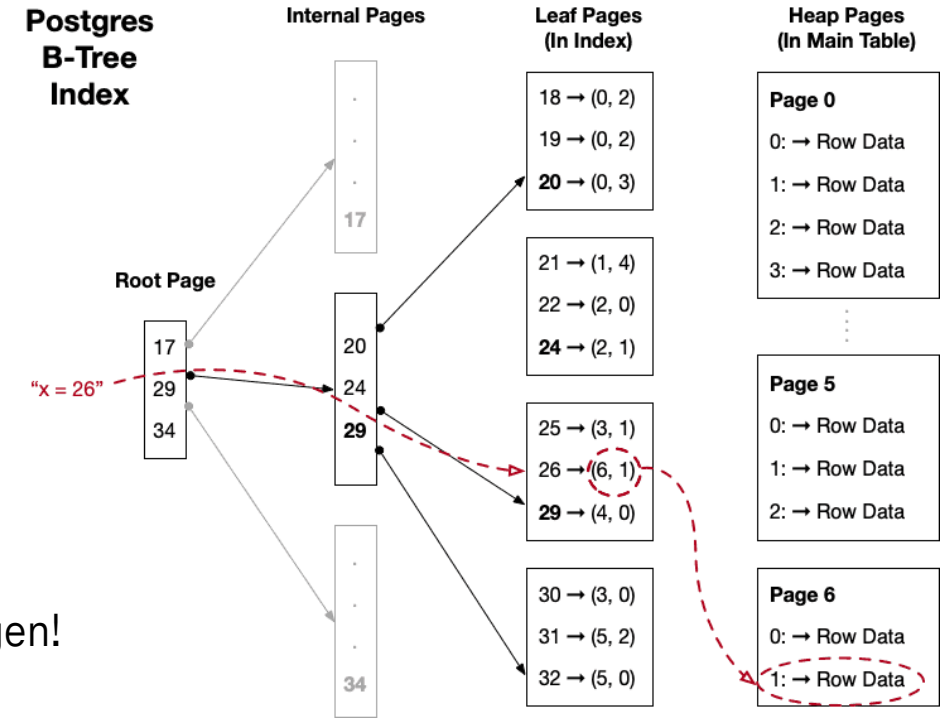
Immer mein erster Schritt!

Vorgehen: Konzentration auf Top 3 Queries, Indexe erstellen, wiederholen bis Erfolgreich

An welchen Stellen besonders auf Indexe achten:

- Foreign Keys
- Constraints
- Joins
- Conditions (WHERE, GROUP BY, ORDER BY)
- Updates

Ergebnis: Ein besserer Index ändert die Performance um Größenordnungen!



https://pganalyze.com/blog/postgresql-vs-sql-server-btree-index-deduplication

CREATE INDEX ...

```
employees=# \d employees.employee
                                    Table "employees.employee"
   Column   |            Type             | Collation | Nullable |                     Default
------------+-----------------------------+-----------+----------+-----------------------------------
 id         | bigint                      |           | not null | nextval('employees.id_employee_seq')
 birth_date | date                        |           | not null |
 first_name | character varying(14)       |           | not null |
 last_name  | character varying(16)       |           | not null |
 gender     | employees.employee_gender |           | not null |
 hire_date  | date                        |           | not null |
Indexes:
    "idx_16988_primary" PRIMARY KEY, btree (id)
Referenced by:
    TABLE "employees.department_employee" CONSTRAINT "dept_emp_ibfk_1" FOREIGN KEY (employee_id)
        REFERENCES employees.employee(id) ON UPDATE RESTRICT ON DELETE CASCADE
    TABLE "employees.department_manager" CONSTRAINT "dept_manager_ibfk_1" FOREIGN KEY (employee_id)
        REFERENCES employees.employee(id) ON UPDATE RESTRICT ON DELETE CASCADE
    TABLE "employees.salary" CONSTRAINT "salaries_ibfk_1" FOREIGN KEY (employee_id)
        REFERENCES employees.employee(id) ON UPDATE RESTRICT ON DELETE CASCADE
    TABLE "employees.title" CONSTRAINT "titles_ibfk_1" FOREIGN KEY (employee_id)
        REFERENCES employees.employee(id) ON UPDATE RESTRICT ON DELETE CASCADE
```

CREATE INDEX …

```
employees=# EXPLAIN ANALYZE SELECT first_name, last_name FROM employee WHERE first_name LIKE 'Martin';

                                                          QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Gather  (cost=1000.00..5733.36 rows=233 width=15) (actual time=16.598..18.827 rows=0.00 loops=1)
   Workers Planned: 1
   Workers Launched: 1
   Buffers: shared hit=2504
   ->  Parallel Seq Scan on employee  (cost=0.00..4710.06 rows=137 width=15) (actual
time=13.756..13.757 rows=0.00 loops=2)
         Filter: ((first_name)::text ~~ 'Martin'::text)
         Rows Removed by Filter: 150012
         Buffers: shared hit=2504
 Planning Time: 0.065 ms
 Execution Time: 18.874 ms
(10 rows)
```

CREATE INDEX ...

```
employees=# CREATE INDEX emp_first_name ON employees.employee USING BTREE (first_name);
CREATE INDEX
employees=# EXPLAIN ANALYZE SELECT first_name, last_name FROM employee WHERE first_name LIKE 'Martin';
                                                      QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on employee  (cost=6.23..701.49 rows=233 width=15) (actual time=0.127..0.127
rows=0.00 loops=1)
   Filter: ((first_name)::text ~~ 'Martin'::text)
   Buffers: shared read=3
   ->  Bitmap Index Scan on emp_first_name  (cost=0.00..6.17 rows=233 width=0) (actual
time=0.121..0.121 rows=0.00 loops=1)
         Index Cond: ((first_name)::text = 'Martin'::text)
         Index Searches: 1
         Buffers: shared read=3
 Planning:
   Buffers: shared hit=16 read=1
 Planning Time: 0.443 ms
 Execution Time: 0.275 ms
(11 rows)
```

Ich glaube nicht an Indexe!

Ein Index verändert das Resultat der Abfrage!

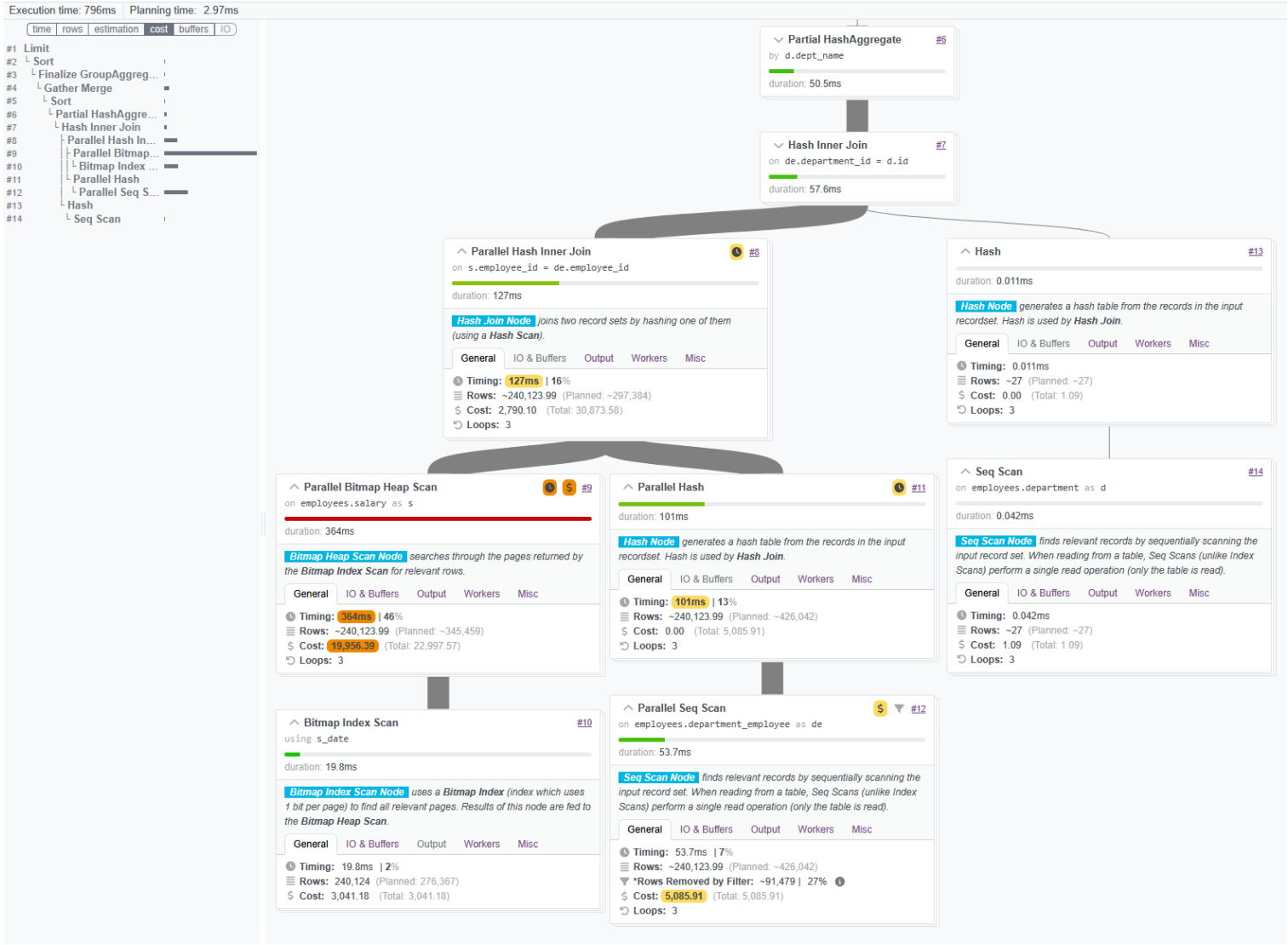
Durch zu viele Indexe wird das Schreiben langsamer!

CREATE INDEX …

```
employees=# EXPLAIN ANALYZE SELECT d.dept_name, AVG(s.amount) AS average_salary
FROM employees.salary s
JOIN employees.department_employee de ON s.employee_id = de.employee_id
JOIN employees.department d ON de.department_id = d.id
WHERE s.to_date > CURRENT_DATE AND de.to_date > CURRENT_DATE
GROUP BY d.dept_name
ORDER BY average_salary DESC
LIMIT 5;
```

```
 Planning Time: 0.879 ms
 Execution Time: 353.408 ms
(51 rows)
```

CREATE INDEX ...



CREATE INDEX ...

Zu beachten:

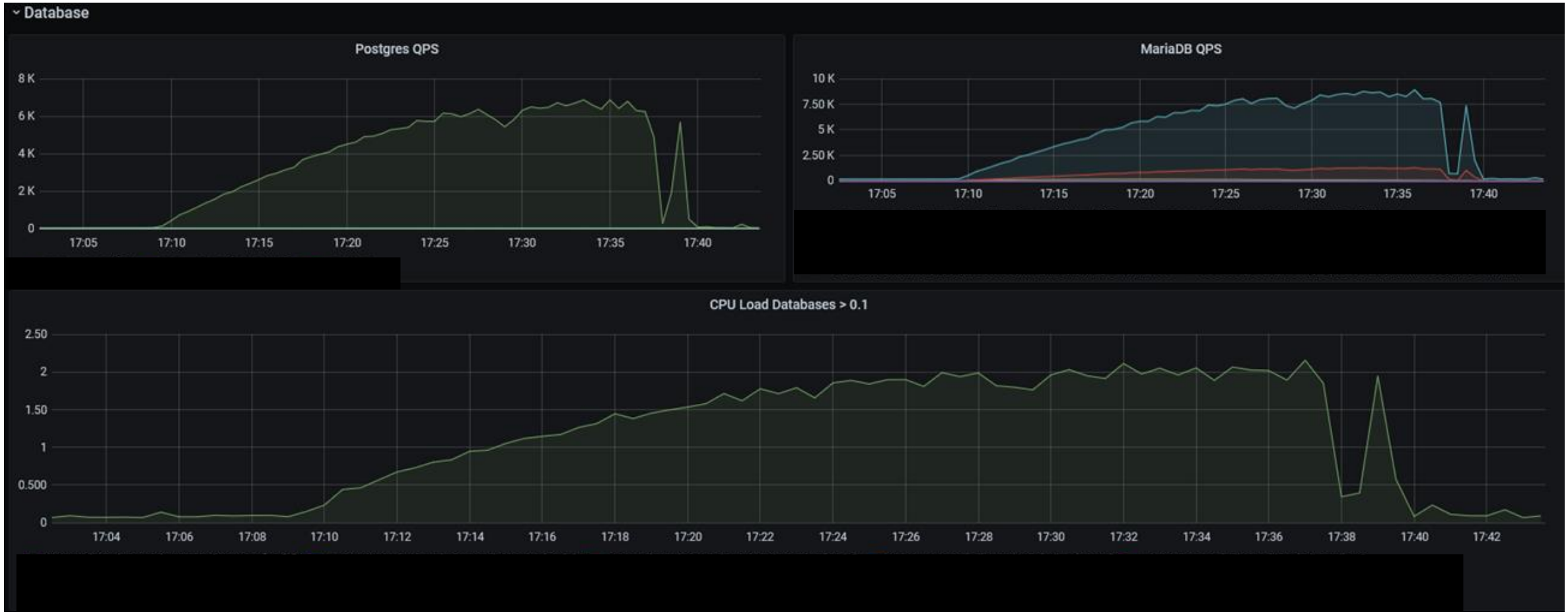
- Redundante Indexe entfernen
- Schlechter Index vs. guter Index
- Kardinalität
- Statistiken können kippen
- Degeneration durch Inserts
- Zeichensätze und Collations

Features:

- Es gibt mehr als BTREE
- Multicolumn Indexe
- Covering Indexe
- Function Indexe
- Queries ohne Index verweigern

ALTER DATABASE production ADD INDEX;

ALTER DATABASE production ADD INDEX;



B, wie

Bytes

INSERT INTO database VALUES („memory“)

Typischerweise der zweite Schritt

Indikatoren: Hohe IO wait oder viel Disk read IO

Rahmenparameter:

- Genügend Arbeitsspeicher allokieren, dass wenige/keine physical Reads stattfinden
- Gilt auch bei aktuellen SSD/NVME Storages!
- Caches, Buffer, Pools, Logs korrekt sizen

Ziel:

- (knappe) IO Ressourcen sind aus dem kritischen Pfad entfernt

Was ist besser als viel Hubraum? – Mehr Hubraum!

INSERT INTO database VALUES („growth“)

Datenbanken und Anwendungen sind länger im Betrieb als erwartet und Wachsen stärker als antizipiert

- Man sieht sich wieder, manchmal nach einem Jahr, manchmal nach einer Woche

Plane für die Zukunft:

- Skalierung der Datenmenge O(n) mit Nutzern
- Historisierte Daten und Logs von OLTP separieren
- Datentypen, 32 Bit können schnell voll sein ;)
- Bereinigungen
  - Fachlich: Wann werden Daten „archiviert“
  - Operativ: DB Maintenance (Vacuum, Index Reorganisation, Partitionierung, Transaktionslogs)

B, wie

Besser machen

CREATE VIEW ideas AS SELECT * FROM experience;

Genereller Umgang mit Datenbanken im Projekt

- Mehrere verschiedene DBMS, aber nicht ALLE
- Optimierung und Budget muss verhältnismäßig sein
- Bilder/Graphen machen Sachverhalte begreifbar
- Buy-In über fachliche Zusammenhänge erzeugen

Entwicklungsprozess

- Schemaversionierung
- ORM
  - Abstraktion vs. Optimierung
  - Lazy Loading
  - Pagination
  - Frustquelle!

Maintenance

- Komplexe HA Lösungen vermeiden
- Recovery planen!
  - **Das Recovery auch testen!!!**
- Performance Baselines sichern
- Feedback für Performancereviews aufbauen
- „Alle haben ein Testsystem, manche gönnen sich auch ein separates Produktionssystem“

COMMIT;

Was wollte ich vermitteln:

- Empirisches Vorgehen ist bei Optimierungen wichtig
- Fokus vs. Rauschen
- Indexe sind das oft vergessene Mittel zum Ziel
- An Business Kontext orientieren
- Einfachheit siegt! (Occams Razor)
- Monitoring dauerhaft betrachten

Toolempfehlung

- PMM
- coroot
- Netdata
- Dalibo PEV2
- pg_stat_statements
- Liquibase

Wissen

- https://www.percona.com/blog/