Optimierung mit Index:

Query 2:

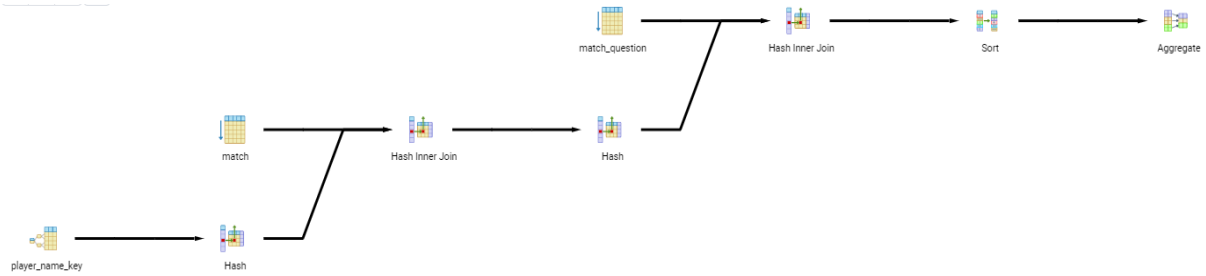
ohne Index:

```
explain analyze select m.id, m.matchstart, m.matchend, count(*) as questions, m.numberofcorrectanswers
from myschema.Player p, myschema.Match m
inner join myschema.match_question m_q on m_q.match_id = m.id
where p.name = 'Beti'
and p.id = m.playinggamer_id
group by m.id
```

Data Output | Explain | Messages | Notifications

QUERY PLAN
text

```
GroupAggregate  (cost=81.43..81.71 rows=16 width=24) (actual time=0.060..0.061 rows=2 loops=1)
  Group Key: m.id
  -> Sort  (cost=81.43..81.47 rows=16 width=16) (actual time=0.057..0.058 rows=7 loops=1)
        Sort Key: m.id
        Sort Method: quicksort  Memory: 25kB
        -> Hash Join  (cost=39.88..81.11 rows=16 width=16) (actual time=0.050..0.052 rows=7 loops=1)
              Hash Cond: (m_q.match_id = m.id)
              -> Seq Scan on match_question m_q  (cost=0.00..32.60 rows=2260 width=4) (actual time=0.007..0.008 rows=7 loops=1)
              -> Hash  (cost=39.73..39.73 rows=12 width=16) (actual time=0.026..0.026 rows=2 loops=1)
                    Buckets: 1024  Batches: 1  Memory Usage: 9kB
                    -> Hash Join  (cost=8.17..39.73 rows=12 width=16) (actual time=0.023..0.025 rows=2 loops=1)
                          Hash Cond: (m.playinggamer_id = p.id)
                          -> Seq Scan on match m  (cost=0.00..27.00 rows=1700 width=20) (actual time=0.004..0.004 rows=2 loops=1)
                          -> Hash  (cost=8.16..8.16 rows=1 width=4) (actual time=0.013..0.013 rows=1 loops=1)
                                Buckets: 1024  Batches: 1  Memory Usage: 9kB
                                -> Index Scan using player_name_key on player p  (cost=0.14..8.16 rows=1 width=4) (actual time=0.009..0.010 rows=1 loops=1)
                                      Index Cond: ((name)::text = 'Beti'::text)
Planning Time: 0.186 ms
Execution Time: 0.124 ms
```



Beobachtung:
Query braucht sehr lange und hat mehrere Operationen auf der Datenbank.

Query 2 mit Index:

Versuch 1:

```
create index name_fk on myschema.Player (name);
```

Data Output | Explain | Messages | Notifications

```
CREATE INDEX

Query returned successfully in 290 msec.
```

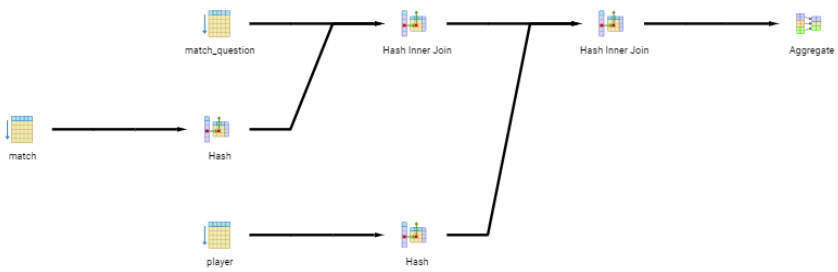
-> Index anlegen

```
explain analyze select m.id, m.matchstart, m.matchend, count(*) as questions, m.
from myschema.Player p, myschema.Match m
inner join myschema.match_question m_q on m_q.match_id = m.id
where p.name = 'Beti'
and p.id = m.playinggamer_id
group by m.id
```

Data Output | Explain | Messages | Notifications

QUERY PLAN
text

```
HashAggregate  (cost=99.54..110.84 rows=1130 width=24) (actual time=0.171..0.191 rows=2 loops=1)
  Group Key: m.id
  -> Hash Join  (cost=49.29..93.89 rows=1130 width=16) (actual time=0.143..0.157 rows=7 loops=1)
        Hash Cond: (m.playinggamer_id = p.id)
        -> Hash Join  (cost=48.25..86.79 rows=2260 width=20) (actual time=0.068..0.077 rows=7 loops=1)
              Hash Cond: (m_q.match_id = m.id)
              -> Seq Scan on match_question m_q  (cost=0.00..32.60 rows=2260 width=4) (actual time=0.015..0.017 rows=7 loops=1)
              -> Hash  (cost=27.00..27.00 rows=1700 width=20) (actual time=0.024..0.024 rows=2 loops=1)
                    Buckets: 2048  Batches: 1  Memory Usage: 17kB
                    -> Seq Scan on match m  (cost=0.00..27.00 rows=1700 width=20) (actual time=0.015..0.016 rows=2 loops=1)
        -> Hash  (cost=1.02..1.02 rows=1 width=4) (actual time=0.052..0.052 rows=1 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 9kB
              -> Seq Scan on player p  (cost=0.00..1.02 rows=1 width=4) (actual time=0.037..0.040 rows=1 loops=1)
                    Filter: ((name)::text = 'Beti'::text)
                    Rows Removed by Filter: 1
Planning Time: 0.723 ms
Execution Time: 0.552 ms
```



Beobachtung:
Braucht mit Index länger. Tabelle vor Aggregate ist sehr lang, benötigt viel Bearbeitungsaufwand.

Versuch 2:

```
1  create index id_fk on myschema.match (id);
```

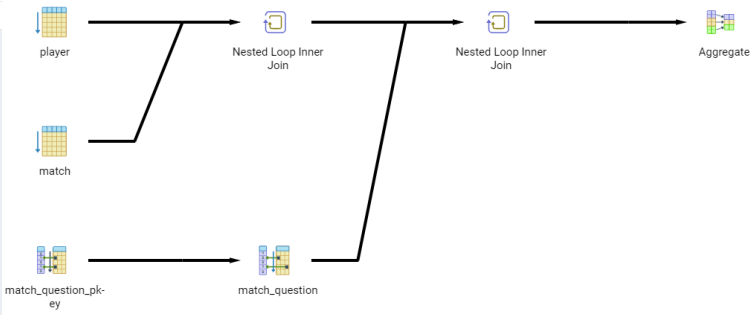
```
1  explain analyze select m.id, m.matchstart, m.matchend, count(*) as questions, m.numberofcorrectanswers
2  from myschema.Player p, myschema.Match m
3  inner join myschema.match_question m_q on m_q.match_id = m.id
4  where p.name = 'Beti'
5  and p.id = m.playinggamer_id
6  group by m.id
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	HashAggregate (cost=38.47..38.49 rows=2 width=24) (actual time=0.033..0.034 rows=2 loops=1)
2	Group Key: m.id
3	-> Nested Loop (cost=21.11..32.82 rows=1130 width=16) (actual time=0.022..0.028 rows=7 loops=1)
4	-> Nested Loop (cost=0.00..2.07 rows=1 width=16) (actual time=0.013..0.014 rows=2 loops=1)
5	Join Filter: (p.id = m.playinggamer_id)
6	-> Seq Scan on player p (cost=0.00..1.02 rows=1 width=4) (actual time=0.007..0.007 rows=1 loops=1)
7	Filter: ((name)::text = 'Beti'::text)
8	Rows Removed by Filter: 1
9	-> Seq Scan on match m (cost=0.00..1.02 rows=2 width=20) (actual time=0.004..0.005 rows=2 loops=1)
10	-> Bitmap Heap Scan on match_question m_q (cost=21.11..30.64 rows=11 width=4) (actual time=0.004..0.004 rows=4 loops=2)
11	Recheck Cond: (match_id = m.id)
12	Heap Blocks: exact=2
13	-> Bitmap Index Scan on match_question_pkey (cost=0.00..21.11 rows=11 width=0) (actual time=0.002..0.002 rows=4 loops=2)
14	Index Cond: (match_id = m.id)
15	Planning Time: 0.193 ms
16	Execution Time: 0.083 ms

```
1  select m.id, m.matchstart, m.matchend, count(*) as questions, m.numberofcorrectanswers
2  from myschema.Player p, myschema.Match m
3  inner join myschema.match_question m_q on m_q.match_id = m.id
4  where p.name = 'Beti'
5  and p.id = m.playinggamer_id
6  group by m.id
```

Data Output | Explain | Messages | Notifications



Beobachtung:

Durch Nested Loop Operationen ist der Bearbeitungsaufwand geringer als mit hash joins. Die kleineren Tabellen werden als erstes verarbeitet, dadurch verringert sich zusätzlich der Aufwand der Query.

Versuch 2 – Verbesserung:

```
2  create index playinggamer_id_fk on myschema.match (playinggamer_id)
```

```
1  explain analyze select m.id, m.matchstart, m.matchend, count(*) as questions, m.numberofcorrectanswers
2  from myschema.Player p, myschema.Match m
3  inner join myschema.match_question m_q on m_q.match_id = m.id
4  where p.name = 'Beti'
5  and p.id = m.playinggamer_id
6  group by m.id
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	HashAggregate (cost=38.47..38.49 rows=2 width=24) (actual time=0.032..0.032 rows=2 loops=1)
2	Group Key: m.id
3	-> Nested Loop (cost=21.11..32.82 rows=1130 width=16) (actual time=0.021..0.027 rows=7 loops=1)
4	-> Nested Loop (cost=0.00..2.07 rows=1 width=16) (actual time=0.012..0.014 rows=2 loops=1)
5	Join Filter: (p.id = m.playinggamer_id)
6	-> Seq Scan on player p (cost=0.00..1.02 rows=1 width=4) (actual time=0.007..0.007 rows=1 loops=1)
7	Filter: ((name)::text = 'Beti'::text)
8	Rows Removed by Filter: 1
9	-> Seq Scan on match m (cost=0.00..1.02 rows=2 width=20) (actual time=0.004..0.005 rows=2 loops=1)
10	-> Bitmap Heap Scan on match_question m_q (cost=21.11..30.64 rows=11 width=4) (actual time=0.003..0.004 rows=4 loops=2)
11	Recheck Cond: (match_id = m.id)
12	Heap Blocks: exact=2
13	-> Bitmap Index Scan on match_question_pkey (cost=0.00..21.11 rows=11 width=0) (actual time=0.002..0.002 rows=4 loops=2)
14	Index Cond: (match_id = m.id)
15	Planning Time: 0.216 ms
16	Execution Time: 0.072 ms

-> Explain-Ansicht unverändert

Beobachtung: leichte Verbesserung der Bearbeitungszeit

Alle Indexe:

```
1  select * from pg_indexes where schemaname = 'myschema';
```

Data Output | Explain | Messages | Notifications

	schemaname name	tablename name	indexname name	tablespace name	indexdef text
1	myschema	player	player_pkey	[null]	CREATE UNIQUE INDEX player_pkey ON myschema.player USING btree (id)
2	myschema	player	player_name_key	[null]	CREATE UNIQUE INDEX player_name_key ON myschema.player USING btree (name)
3	myschema	category	category_pkey	[null]	CREATE UNIQUE INDEX category_pkey ON myschema.category USING btree (id)
4	myschema	category	category_name_key	[null]	CREATE UNIQUE INDEX category_name_key ON myschema.category USING btree (name)
5	myschema	question	question_pkey	[null]	CREATE UNIQUE INDEX question_pkey ON myschema.question USING btree (id)
6	myschema	match	match_pkey	[null]	CREATE UNIQUE INDEX match_pkey ON myschema.match USING btree (id)
7	myschema	match_categ...	match_category_pkey	[null]	CREATE UNIQUE INDEX match_category_pkey ON myschema.match_category USING btree (category_id, match_id)
8	myschema	match_quest...	match_question_pkey	[null]	CREATE UNIQUE INDEX match_question_pkey ON myschema.match_question USING btree (question_id, match_id)
9	myschema	match	id_f	[null]	CREATE INDEX id_f ON myschema.match USING btree (id)
10	myschema	match	playinggamer_id_fk	[null]	CREATE INDEX playinggamer_id_fk ON myschema.match USING btree (playinggamer_id)

Query 1 - Optimierung

```
explain analyze SELECT distinct p.name
    FROM myschema.Player p, myschema.Match m
    where p.id = m.playinggamer_id
    and m.matchstart between '2019-06-01' and '2019-06-20'
```

Data Output | Explain | Messages | Notifications

```
QUERY PLAN
text
Unique  (cost=47.70..47.75 rows=8 width=516) (actual time=0.122..0.123 rows=1 loops=1)
 ->  Sort  (cost=47.70..47.73 rows=8 width=516) (actual time=0.122..0.122 rows=1 loops=1)
       Sort Key: p.name
       Sort Method: quicksort  Memory: 25kB
       ->  Hash Join  (cost=35.60..47.59 rows=8 width=516) (actual time=0.059..0.060 rows=1 loops=1)
             Hash Cond: (p.id = m.playinggamer_id)
             ->  Seq Scan on player p  (cost=0.00..11.40 rows=140 width=520) (actual time=0.024..0.024 rows=2 loops=1)
             ->  Hash  (cost=35.50..35.50 rows=8 width=4) (actual time=0.023..0.023 rows=1 loops=1)
                   Buckets: 1024  Batches: 1  Memory Usage: 9kB
                   ->  Seq Scan on match m  (cost=0.00..35.50 rows=8 width=4) (actual time=0.019..0.019 rows=1 loops=1)
                         Filter: ((matchstart >= '2019-06-01'::date) AND (matchstart <= '2019-06-20'::date))
Planning Time: 0.175 ms
Execution Time: 0.156 ms
```

-> ohne Indexe

```
explain analyze SELECT distinct p.name
    FROM myschema.Player p, myschema.Match m
    where p.id = m.playinggamer_id
    and m.matchstart between '2019-06-01' and '2019-06-20'
```

Data Output | Explain | Messages | Notifications

```
QUERY PLAN
text
Unique  (cost=9.30..9.31 rows=1 width=516) (actual time=0.036..0.036 rows=1 loops=1)
 ->  Sort  (cost=9.30..9.31 rows=1 width=516) (actual time=0.036..0.036 rows=1 loops=1)
       Sort Key: p.name
       Sort Method: quicksort  Memory: 25kB
       ->  Nested Loop  (cost=0.14..9.29 rows=1 width=516) (actual time=0.028..0.028 rows=1 loops=1)
             ->  Seq Scan on match m  (cost=0.00..1.01 rows=1 width=4) (actual time=0.010..0.010 rows=1 loops=1)
                   Filter: ((matchstart >= '2019-06-01'::date) AND (matchstart <= '2019-06-20'::date))
             ->  Index Scan using player_pkey on player p  (cost=0.14..8.16 rows=1 width=520) (actual time=0.016..0.016 rows=1 loops=1)
                   Index Cond: (id = m.playinggamer_id)
Planning Time: 0.131 ms
Execution Time: 0.058 ms
```

-> mit Indexe

Query 3 - Optimierung

```
explain analyze  select p.name, count(*) as matchnumber
from myschema.Player p
inner join myschema.Match m on p.id = m.playingGamer_id
group by p.name
```

Data Output | Explain | Messages | Notifications

```
QUERY PLAN
text
Sort  (cost=59.59..59.94 rows=140 width=524) (actual time=0.048..0.048 rows=1 loops=1)
  Sort Key: (count(*)) DESC
  Sort Method: quicksort  Memory: 25kB
  ->  HashAggregate  (cost=53.20..54.60 rows=140 width=524) (actual time=0.034..0.035 rows=1 loops=1)
        Group Key: p.name
        ->  Hash Join  (cost=13.15..44.70 rows=1700 width=516) (actual time=0.029..0.030 rows=1 loops=1)
              Hash Cond: (m.playinggamer_id = p.id)
              ->  Seq Scan on match m  (cost=0.00..27.00 rows=1700 width=4) (actual time=0.009..0.010 rows=1 loops=1)
              ->  Hash  (cost=11.40..11.40 rows=140 width=520) (actual time=0.011..0.011 rows=2 loops=1)
                    Buckets: 1024  Batches: 1  Memory Usage: 9kB
                    ->  Seq Scan on player p  (cost=0.00..11.40 rows=140 width=520) (actual time=0.006..0.007 rows=2 loops=1)
Planning Time: 0.163 ms
Execution Time: 0.095 ms
```

-> ohne Indexe

```
explain analyze select p.name, count(*) as matchnumber
from myschema.Player p
inner join myschema.Match m on p.id = m.playingGamer_id
group by p.name
order by count(*) DESC
```

Data Output | Explain | Messages | Notifications

```
QUERY PLAN
text
Sort  (cost=9.33..9.33 rows=1 width=524) (actual time=0.046..0.046 rows=1 loops=1)
  Sort Key: (count(*)) DESC
  Sort Method: quicksort  Memory: 25kB
  ->  GroupAggregate  (cost=9.30..9.32 rows=1 width=524) (actual time=0.043..0.043 rows=1 loops=1)
        Group Key: p.name
        ->  Sort  (cost=9.30..9.30 rows=1 width=516) (actual time=0.040..0.040 rows=1 loops=1)
              Sort Key: p.name
              Sort Method: quicksort  Memory: 25kB
              ->  Nested Loop  (cost=0.14..9.29 rows=1 width=516) (actual time=0.034..0.035 rows=1 loops=1)
                    ->  Seq Scan on match m  (cost=0.00..1.01 rows=1 width=4) (actual time=0.020..0.020 rows=1 loops=1)
                    ->  Index Scan using player_pkey on player p  (cost=0.14..8.16 rows=1 width=520) (actual time=0.011..0.011 rows=1 loops=1)
                          Index Cond: (id = m.playinggamer_id)
Planning Time: 0.164 ms
Execution Time: 0.084 ms
```

-> mit Indexe

Query 4 - Optimierung

```
explain analyze SELECT c.name, count(m_c.category_id) as callNumber
FROM myschema.category c
inner join myschema.match_category m_c on c.id = m_c.category_id
GROUP BY c.name
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Sort (cost=54.06..54.19 rows=51 width=18) (actual time=0.070..0.070 rows=2 loops=1)
2	Sort Key: (count(m_c.category_id)) DESC
3	Sort Method: quicksort Memory: 25kB
4	-> HashAggregate (cost=52.10..52.61 rows=51 width=18) (actual time=0.065..0.066 rows=2 loops=1)
5	Group Key: c.name
6	-> Hash Join (cost=2.15..40.80 rows=2260 width=14) (actual time=0.061..0.062 rows=2 loops=1)
7	Hash Cond: (m_c.category_id = c.id)
8	-> Seq Scan on match_category m_c (cost=0.00..32.60 rows=2260 width=4) (actual time=0.013..0.013 rows=2 loops=1)
9	-> Hash (cost=1.51..1.51 rows=51 width=14) (actual time=0.043..0.043 rows=51 loops=1)
10	Buckets: 1024 Batches: 1 Memory Usage: 11kB
11	-> Seq Scan on category c (cost=0.00..1.51 rows=51 width=14) (actual time=0.010..0.016 rows=51 loops=1)
12	Planning Time: 2.075 ms
13	Execution Time: 0.106 ms

-> ohne Indexe

```
explain analyze SELECT c.name, count(m_c.category_id) as callNumber
FROM myschema.category c
inner join myschema.match_category m_c on c.id = m_c.category_id
GROUP BY c.name
order by count(m_c.category_id) DESC
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Sort (cost=54.06..54.19 rows=51 width=18) (actual time=0.051..0.051 rows=2 loops=1)
2	Sort Key: (count(m_c.category_id)) DESC
3	Sort Method: quicksort Memory: 25kB
4	-> HashAggregate (cost=52.10..52.61 rows=51 width=18) (actual time=0.043..0.044 rows=2 loops=1)
5	Group Key: c.name
6	-> Hash Join (cost=2.15..40.80 rows=2260 width=14) (actual time=0.039..0.040 rows=2 loops=1)
7	Hash Cond: (m_c.category_id = c.id)
8	-> Seq Scan on match_category m_c (cost=0.00..32.60 rows=2260 width=4) (actual time=0.009..0.010 rows=2 loops=1)
9	-> Hash (cost=1.51..1.51 rows=51 width=14) (actual time=0.024..0.024 rows=51 loops=1)
10	Buckets: 1024 Batches: 1 Memory Usage: 11kB
11	-> Seq Scan on category c (cost=0.00..1.51 rows=51 width=14) (actual time=0.008..0.012 rows=51 loops=1)
12	Planning Time: 0.135 ms
13	Execution Time: 0.082 ms

-> mit Indexe