Filtering with WHERE

IMPROVING QUERY PERFORMANCE IN SQL SERVER

Dean Smith
Founder, Atamai Analytics

How WHERE works

```
SELECT *
FROM PlayerStats
WHERE Position = 'SG'
```

PlayerName	Team	Position	GamesPlayed	MinutesPlayed	ORebound	DRebound	Assists	Steals	Blocks	TurnOvers	TotalPoints
Luke Babbitt	MIA	SF	13	145	2	13	5	1	2	4	33
Dwayne Bacon	CHO	SG	53	713	4	120	38	16	2	23	175
Ron Baker	NYK	SG	29	385	5	25	47	26	6	18	71
Wade Baldwin	POR	PG	7	80	1	7	5	2	1	4	38
Lonzo Ball	LAL	PG	52	1780	69	291	376	88	43	136	528
J.J. Barea	DAL	PG	69	1603	15	186	434	35	3	143	801
Harrison Barnes	DAL	PF	77	2634	77	391	152	49	14	118	1452
Will Barton	DEN	SG	81	2683	70	339	331	82	52	149	1268
DeAndre' Bembry	ATL	SF	26	455	8	64	49	21	12	47	136

How WHERE works

```
SELECT *
FROM PlayerStats
WHERE Position = 'SG'
```

PlayerName	Team	Position	GamesPlayed	MinutesPlayed	ORebound	DRebound	Assists	Steals	Blocks	TurnOvers	TotalPoints
Luke Babbitt	MIA	SF	13	145	2	13	5	1	2	4	33
Dwayne Bacon	CHO	SG	53	713	4	120	38	16	2	23	175
Ron Baker	NYK	SG	29	385	5	25	47	26	6	18	71
Wade Baldwin	POR	PG	7	80	1	7	5	2	1	4	38
Lonzo Ball	LAL	PG	52	1780	69	291	376	88	43	136	528
J.J. Barea	DAL	PG	69	1603	15	186	434	35	3	143	801
Harrison Barnes	DAL	PF	77	2634	77	391	152	49	14	118	1452
Will Barton	DEN	SG	81	2683	70	339	331	82	52	149	1268
DeAndre' Bembry	ATL	SF	26	455	8	64	49	21	12	47	136

WHERE processing order

```
SELECT PlayerName,
      Team,
      (DRebound+ORebound) AS TotalRebounds
FROM PlayerStats
WHERE TotalRebounds >= 1000
ORDER BY TotalRebounds DESC;
```

```
-- ERROR
Invalid column name 'TotalRebounds'.
```

- **WHERE** is processed before **SELECT**

Using a sub-query

```
SELECT PlayerName,
       Team,
       TotalRebounds
FROM
     -- Start of sub-query tr
    (SELECT PlayerName, Team,
            (DRebound+ORebound) AS TotalRebounds
     FROM PlayerStats) tr
WHERE TotalRebounds >= 1000 -- created in the sub-query
ORDER BY TotalRebounds DESC;
```

Using a sub-query

```
SELECT PlayerName,
       Team,
       TotalRebounds
FROM
     -- Start of sub-query tr
    (SELECT PlayerName, Team,
            (DRebound+ORebound) AS TotalRebounds
     FROM PlayerStats) tr
WHERE TotalRebounds >= 1000 -- created in the sub-query
ORDER BY TotalRebounds DESC;
```

PlayerName	Team	TotalRebounds
Andre Drummond	DET	1247
DeAndre Jordan	LAC	1171
Karl-Anthony Towns	MIN	1012
Dwight Howard	CHO	1012

Calculations on columns

```
SELECT PlayerName,
       Team,
       (DRebound+ORebound) AS TotalRebounds
FROM PlayerStats
WHERE (DRebound+ORebound) >= 1000
ORDER BY TotalRebounds DESC;
```

- *Calculations* on columns in the **WHERE** filter condition could increase query times

PlayerName	Team	TotalRebounds
Andre Drummond	DET	1247
DeAndre Jordan	LAC	1171
Karl-Anthony Towns	MIN	1012
Dwight Howard	CHO	1012

Functions on columns

```
SELECT PlayerName, College, DraftYear
FROM Players
WHERE UPPER(LEFT(College,7)) = 'GEORGIA';
-- unnecessary use of functions
-- on a filtering column
```

- Applying *functions* to columns in the **WHERE** filter condition could increase query times

PlayerName	College	DraftYear
Damien Wilkins	Georgia	
Derrick Favors	Georgia Tech	2010
Iman Shumpert	Georgia Tech	2011
R.J. Hunter	Georgia State	2015
...	...	...

WHERE simplified

```
SELECT PlayerName, College, DraftYear
FROM Players
        -- No calculation or function
WHERE College like 'Georgia%';
```

PlayerName	College	DraftYear
Damien Wilkins	Georgia	
Derrick Favors	Georgia Tech	2010
Iman Shumpert	Georgia Tech	2011
R.J. Hunter	Georgia State	2015
...	...	...

Summary

- **WHERE** is processed before **SELECT**
- *Calculations* on columns in the **WHERE** filter condition could increase query times
- Applying *functions* to columns in the **WHERE** filter condition could increase query times

Let's practice!

IMPROVING QUERY PERFORMANCE IN SQL SERVER

Filtering with HAVING

IMPROVING QUERY PERFORMANCE IN SQL SERVER

Dean Smith
Founder, Atamai Analytics

HAVING processing order

1. FROM

4. WHERE

6. HAVING

7. SELECT

Grouping with WHERE row filter

```
SELECT Team,
    SUM(TotalPoints) AS TotalSGTeamPoints
FROM PlayerStats
WHERE Position = 'SG'
GROUP BY Team
```

Team	TotalSGTeamPoints
ATL	2034
BOS	1606
BRK	2126
CHI	2905
CHO	2661
CLE	2489
...	...

Row filtering with HAVING

```
SELECT Team,
    SUM(TotalPoints) AS TotalSGTeamPoints
FROM PlayerStats
WHERE Position = 'SG'
GROUP BY Team
```

```
SELECT Team,
    SUM(TotalPoints) AS TotalSGTeamPoints
FROM PlayerStats
--WHERE Position = 'SG'
GROUP BY Team, Position
HAVING Position = 'SG'
```

Don't use **HAVING** to filter *individual* or *ungrouped* rows

Team	TotalSGTeamPoints
ATL	2034
BOS	1606
BRK	2126
CHI	2905
CHO	2661
CLE	2489
...	...

Aggregating by group

```
SELECT
    Team,
    SUM(DRebound+ORebound) AS TotRebounds,
    SUM(DRebound) AS TotDef,
    SUM(ORebound) AS TotOff
FROM PlayerStats
GROUP BY Team;
```

Team	TotRebounds	TotDef	TotOff
ATL	3436	2693	743
BOS	3645	2878	767
BRK	3644	2852	792
CHI	3663	2873	790
CHO	3728	2901	827
CLE	3455	2761	694
...	...	...	...

Group filtering with WHERE

```
SELECT
    Team,
    SUM(DRebound+ORebound) AS TotRebounds,
    SUM(DRebound) AS TotDef,
    SUM(ORebound) AS TotOff
FROM PlayerStats
WHERE ORebound >= 1000
GROUP BY Team;
```

Team	TotRebounds	TotDef	TotOff

- Use **WHERE** to filter individual rows and **HAVING** for a numeric filter on *grouped* rows

Without an aggregate function

```
SELECT
    Team,
    SUM(DRebound+ORebound) AS TotRebounds,
    SUM(DRebound) AS TotDef,
    SUM(ORebound) AS TotOff
FROM PlayerStats
GROUP BY Team
HAVING ORebound >= 1000;
```

```
-------------------------------------------------
-- ERROR
Column 'PlayerStats.ORebound' is invalid in the
HAVING clause because it is not contained in
either an aggregate function or the GROUP BY
clause.
```

- Apply the **HAVING** filter to a numeric column using an aggregate function

With an aggregate function

```
SELECT
    Team,
    SUM(DRebound+ORebound) AS TotRebounds,
    SUM(DRebound) AS TotDef,
    SUM(ORebound) AS TotalOff
FROM PlayerStats
GROUP BY Team
    -- aggregate function SUM()
HAVING SUM(ORebound) >= 1000;
```

Team	TotRebounds	TotDef	TotOff
OKC	3695	2671	1024

Summary

- Do not use **HAVING** to filter individual or ungrouped rows
- Use **WHERE** to filter individual rows and **HAVING** for a numeric filter on *grouped* rows
- **HAVING** can only be applied to a numeric column in an aggregate function filter

Let's practice!

IMPROVING QUERY PERFORMANCE IN SQL SERVER

Interrogation after SELECT

IMPROVING QUERY PERFORMANCE IN SQL SERVER

Dean Smith
Founder, Atamai Analytics

Processing order after SELECT

1. FROM
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. HAVING

7. SELECT

8. DISTINCT
9. ORDER BY
10. TOP

All is not always good

```
SELECT *
FROM PlayerStats;
```

`SELECT *` is great for data interrogation but potentially bad for performance

PlayerName	Team	Position	...	...
Alex Abrines	OKC	SG	...	...
Quincy Acy	BRK	PF	...	...
Steven Adams	OKC	C	...	...
Bam Adebayo	MIA	C	...	...
...	...	...	...	...

All you need is better

```
SELECT PlayerName,
       Team,
       TotalPoints
FROM PlayerStats;
```

Only select the columns required

PlayerName	Team	TotalPoints
Alex Abrines	OKC	353
Quincy Acy	BRK	411
Steven Adams	OKC	1056
Bam Adebayo	MIA	477
...	...	...

All in a JOIN

```
SELECT *
FROM Players p
  INNER JOIN PlayerStats ps
  ON ps.PlayerName = p.PlayerName;
```

`SELECT *` in joins returns duplicates of joining columns

PlayerName	Age	PlayerName	Team	...
Alex Abrines	24	Alex Abrines	OKC	...
Quincy Acy	27	Quincy Acy	BRK	...
Steven Adams	24	Steven Adams	OKC	...
Bam Adebayo	20	Bam Adebayo	MIA	...
...	...	...	...	...

All you need in a JOIN

```
SELECT p.PlayerName,
       ps.Team,
       p.Country,
       ps.Position
FROM Players p
  INNER JOIN PlayerStats ps
  ON ps.PlayerName = p.PlayerName;
```

Explicitly state the columns to be returned and from what tables

PlayerName	Country	Team	Position
Alex Abrines	Spain	OKC	SG
Quincy Acy	USA	BRK	PF
Steven Adams	New Zealand	OKC	C
Bam Adebayo	USA	MIA	C
...	...	...	...

Rows at the TOP

```
SELECT TOP 5 PlayerName,
        Team,
        TotalPoints
FROM PlayerStats;
```

PlayerName	Team	TotalPoints
Alex Abrines	OKC	353
Quincy Acy	BRK	411
Steven Adams	OKC	1056
Bam Adebayo	MIA	477
Arron Afflalo	ORL	179

Percentage at the TOP

```
SELECT TOP 1 PERCENT PlayerName,
        Team,
        TotalPoints
FROM PlayerStats;
```

PlayerName	Team	TotalPoints
Alex Abrines	OKC	353
Quincy Acy	BRK	411
Steven Adams	OKC	1056
Bam Adebayo	MIA	477
Arron Afflalo	ORL	179
Cole Aldrich	MIN	12
LaMarcus Aldridge	SAS	1735

There is no top or bottom

```
SELECT TOP 5 PlayerName,
        Team,
        TotalPoints
FROM PlayerStats
ORDER BY TotalPoints DESC
```

PlayerName	Team	TotalPoints
LeBron James	CLE	2251
James Harden	HOU	2191
Anthony Davis	NOP	2110
Russell Westbrook	OKC	2028
Giannis Antetokounmpo	MIL	2014

The other row limiters

Row limiter	Ordering clause	Database
TOP	ORDER BY	Microsoft SQL Server
ROWNUM	ORDER BY	Oracle
LIMIT	ORDER BY	PostgreSQL

Where to use ORDER BY

Use `ORDER BY` in a query:

- To interrogate the data
- When there is a good reason for the final returned results to be sorted

Let's practice!

IMPROVING QUERY PERFORMANCE IN SQL SERVER

Managing duplicates

IMPROVING QUERY PERFORMANCE IN SQL SERVER

Dean Smith
Founder, Atamai Analytics

Query returning duplicates

```
SELECT PlayerName,
       Team
FROM PlayerStats;
```

PlayerName	Team
...	...
Emmanuel Mudiay	DEN
Emmanuel Mudiay	NYK
Enes Kanter	NYK
Eric Bledsoe	PHO
Eric Bledsoe	MIL
...	...

Removing duplicates with DISTINCT()

```
SELECT DISTINCT(PlayerName)
FROM PlayerStats;
```

PlayerName
...
Emmanuel Mudiay
Enes Kanter
Eric Bledsoe
Eric Gordon
Eric Moreland
...

GROUP BY instead of DISTINCT()

```
SELECT DISTINCT(PlayerName)
FROM PlayerStats;
```

```
SELECT PlayerName
FROM PlayerStats
GROUP BY PlayerName;
```

GROUP BY instead of DISTINCT()

```
SELECT DISTINCT(PlayerName)
FROM PlayerStats;
```

```
SELECT PlayerName
FROM PlayerStats
GROUP BY PlayerName;
```

```
SELECT PlayerName,
    COUNT(Team) AS TeamsPlayedFor
FROM PlayerStats
GROUP BY PlayerName;
```

PlayerName	TeamsPlayedFor
...	...
Emmanuel Mudiay	2
Enes Kanter	1
Eric Bledsoe	2
Eric Gordon	1
Eric Moreland	1
...	...

Is there another way?

```
SELECT PlayerName
FROM Players;
```

PlayerName
...
Emmanuel Mudiay
Enes Kanter
Eric Bledsoe
Eric Gordon
Eric Moreland
...

Fruits tables

Fruits1

FruitName	FruitType
Grapefruit	Citrus
Orange	Citrus
Peach	Stone
Strawberry	Berry

Fruits2

FruitName	FruitType
Marionberry	Berry
Orange	Citrus
Plum	Stone
Strawberry	Berry

Duplicate fruits

- `Orange` and `Strawberry` are duplicated

Fruits1 appended to **Fruits2**

FruitName	FruitType
...	...
Orange	Citrus
Orange	Citrus
Plum	Stone
Strawberry	Berry
Strawberry	Berry

Remove duplicates with UNION

```
SELECT FruitName, FruitType
FROM Fruits1

UNION

SELECT FruitName, FruitType
FROM Fruits2
```

FruitName	FruitType
Grapefruit	Citrus
Orange	Citrus
Peach	Stone
Strawberry	Berry
Marionberry	Berry
Plum	Stone

What about UNION ALL?

- **UNION ALL**
  - appends rows from one or more tables and *does not* remove duplicate rows

- **UNION**
  - appends rows from one or more tables and removes duplicate rows

DISTINCT() and UNION

- Use with caution
- May use an internal sort mechanism to order and check for duplicates
- Potentially increase the time it takes for a query to run

Using DISTINCT()

Before using `DISTINCT()` we should ask:

- Is there an alternative method?
  - using a table with a unique key instead
- Is the query using an aggregate function?
  - group with `GROUP BY`

Using UNION

Before using `UNION` we should ask:

- Are duplicate rows OK?
- Will appending queries produce duplicate rows?
- Consider using `UNION ALL` if duplicate rows are OK or if no duplicate rows will be created

Let's practice!

IMPROVING QUERY PERFORMANCE IN SQL SERVER