Štatistika operácií

Alena Martinková, MAR0702

DAIS 2

1. Štatistiky operácií

Tabuľka Statistic

Kód	Operácia	Četnost	Očakávaná veľkosť výsledku	IO Cost	IO Cost 2	Čas operácie (ms)	Čas operácie 2 (ms)	Veľkosť výsledku (MB)	Veľkosť výsledku 2 (MB)
s1	INSERT INTO *Statistic* VALUES (playerID, teamID, assists, goals)	3	1	7	22	-	-	-	-
s2	UPDATE *Statistic* SET assists = x, goals = y WHERE playerID = z AND teamID = a	10	1	363	6	-	-	-	-
s3	SELECT * FROM *Statistic*	1	100 000	363	2	306	60	2,10	0,001
s4	SELECT * FROM *Statistic* WHERE playerID = x	4	1	363	4	-	-	-	-
s5	SELECT count(statisticsID) FROM *Statistic* WHERE playerID = $p_id and teamID = $new_t_id;	5	1	363	4	-	-	-	-

V tabuľke **Statistic** bol index vytvorený na atribúty *playerID* a *teamID*. Index na atribút *playerID* bol vytvorený pre dotazy **s2, s4** a **s5** a index na atribút *teamID* pre dotazy **s2** a **s5**. Ako môžeme vidieť oba indexy výrazne znížili IO cost a môžeme ich považovať za úspešné.

Pre dotaz **s3** bolo vytvorené stránkovanie, z dôvodu prístupu k veľkému množstvu dát, ktoré by užívateľ nikdy nevyhľadal v celku a taktiež vysokému IO costu. Stránkovanie bolo vytvorené v tvare:

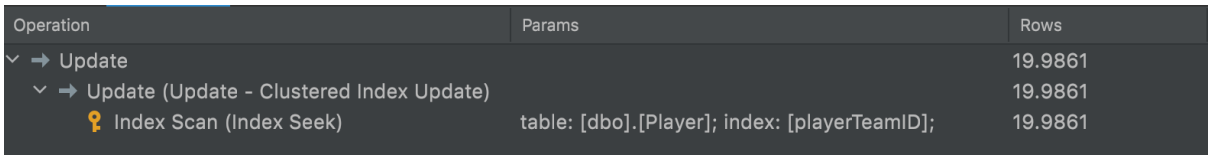
```
DECLARE @PageNumber AS INT, @RowspPage AS INT
SET @PageNumber = 2500
SET @RowspPage = 20
SELECT *
FROM Statistic
ORDER BY statisticsID
OFFSET ((@PageNumber - 1) * @RowspPage) ROWS
FETCH NEXT @RowspPage ROWS ONLY
GO
```

Vidíme, že stránkovanie bolo úspešné, v počte ms, MB aj IO coste.

Tabuľka Player

Kód	Operácia	Četnost	Očakávaná veľkosť výsledku	IO Cost	IO Cost 2	Čas operácie (ms)	Čas operácie 2 (ms)	Veľkosť výsledku (MB)	Veľkosť výsledku 2 (MB)
p1	INSERT INTO *Player* VALUES (teamID, email, dateOfBirth, firstName, lastName, stick, covid)	3	1	5	13	-	-	-	-
p2	UPDATE *Player* SET email = email, lastName = lastName, stick = stick, covid = covid WHERE playerID = x	3	1	3	3	-	-	-	-
p3	SELECT * FROM *Player*	1	199 924	1759	4	6315	62	12	0,002
p4	SELECT * FROM *Player* WHERE teamID = x	5	21	1759	65	-	-	-	-
p5	SELECT * FROM *Player* WHERE playerID = x	5	1	3	3	-	-	-	-
p6	SELECT teamID FROM *Player* WHERE playerID = $p_id;	5	1	3	3	-	-	-	-
p7	SELECT * FROM *Player* WHERE teamID = firstTeamID OR teamID = secondTeamID	5	40	1759	135	-	-	-	-
p8	UPDATE *Player* SET teamID = $new_t_id WHERE playerID = $p_id;	5	1	5	5	-	-	-	-
p9	UPDATE *Player* SET covid = 1 WHERE teamID = $t_id	5	1	1759	62	-	-	-	-

V tabuľke **Player** bol index vytvorený na atribúte *teamID*, z dôvodu kombinácie vysokej selektivity a vysokého IO costu dotazov **p4, p7** a **p9**. Všetkým dotazom sa výrazne znížil IO cost ako môžeme vidieť v tabuľke. Dotaz **p9** má stále IO cost 62, a to z dôvodu, že okrem aktualizácie +-20 záznamov, musí taktiež aktualizovať aj samotný clustered index. Ilustrované na priloženom pláne:



Pre dotaz **p3** bolo vytvorené stránkovanie, z dôvodu prístupu k veľkému množstvu dát, ktoré by užívateľ nikdy nevyhľadal v celku a taktiež vysokému IO costu. Stránkovanie bolo vytvorené v tvare:

```
DECLARE @PageNumber AS INT, @RowspPage AS INT
SET @PageNumber = 5000
SET @RowspPage = 20
SELECT *
FROM Player
ORDER BY playerID
OFFSET ((@PageNumber - 1) * @RowspPage) ROWS
FETCH NEXT @RowspPage ROWS ONLY
GO
```

Vidíme, že stránkovanie bolo úspešné, v počte ms, MB aj IO coste.

Tabuľka Team

Kód	Operácia	Četnost	Očakávaná veľkosť výsledku	IO Cost	IO Cost 2	Čas operácie (ms)	Čas operácie 2 (ms)	Veľkosť výsledku (MB)	Veľkosť výsledku 2 (MB)
t1	INSERT INTO *Team* VALUES (leagueID, name, rank, covid, qurantinedFrom)	3	1	4	11	-	-	-	-
t2	UPDATE *Team* SET leagueID = leagueID, name = name, rank = rank, covid = covid, quarantinedFrom = quarantinedFrom WHERE teamID = x	3	1	4	4	-	-	-	-
t3	SELECT * FROM *Team*	1	10 000	64	2	309	57	0,38	0,001

t4	SELECT * FROM *Team* WHERE leagueID = x	3	10	64	22	-	-	-	-
t5	UPDATE *Team* SET covid = 1, quarantinedFrom = $current_date WHERE teamID = $t_id	5	1	2	2	-	-	-	-
t6	SELECT * FROM *Team* WHERE teamID = x	5	1	2	2	-	-	-	-
t7	SELECT max(rank) FROM *Team* WHERE leagueID = $new_l_id	2	1	64	22	-	-	-	-
t8	SELECT leagueID, rank FROM *Team* WHERE teamID = $t_id	2	1	2	2	-	-	-	-
t9	UPDATE *Team* SET leagueID = $new_l_id, rank = $rank + 1 WHERE teamID = $t_id	2	1	4	4	-	-	-	-
t10	UPDATE *Team* SET rank = rank – 1 WHERE rank >= $old_rank AND rank <= $max_rank AND leagueID = $old_l_id	2	1	64	22	-	-	-	-
t11	SELECT count(teamID) FROM *Team* WHERE leagueID = $l_id	2	1	2	2	-	-	-	-

V tabuľke **Team** bol index vytvorený na atribúte *leagueID* pre zlepšenie IO costu dotazov **t4**, **t7** a **t10**. Pre dotaz **t3** bolo vytvorené stránkovanie, z dôvodu prístupu k veľkému množstvu dát, ktoré by užívateľ nikdy nevyhľadal v celku a taktiež vysokému IO costu. Stránkovanie bolo vytvorené v tvare:

```
DECLARE @PageNumber AS INT, @RowspPage AS INT
SET @PageNumber = 250
SET @RowspPage = 20
SELECT *
FROM Team
ORDER BY teamID
OFFSET ((@PageNumber - 1) * @RowspPage) ROWS
FETCH NEXT @RowspPage ROWS ONLY
GO
```

Vidíme, že stránkovanie bolo úspešné, v počte ms, MB aj IO coste.

Tabuľka League

Kód	Operácia	Četnost	Očakávaná veľkosť výsledku	IO Cost	IO Cost 2	Čas operácie (ms)	Čas operácie 2 (ms)	Veľkosť výsledku (MB)	Veľkosť výsledku 2 (MB)
I1	INSERT INTO *League* VALUES (name, category)	3	1	2	2	-	-	-	-
I2	UPDATE *League* SET name = name, category = category WHERE leagueID = x	2	1	2	2	-	-	-	-
I3	SELECT * FROM *League*	1	1001	6	2	90	57	0,01	0,0005
I4	SELECT * FROM *League* WHERE leagueID = x	4	1	2	2	-	-	-	-

V tabuľke **League** nebol vytvorený žiadny index, pretože na základe údajov z tabulky nie je žiaden potreba. Jedine pre dotaz **I3** bolo vytvorené stránkovanie, z dôvodu prístupu k veľkému množstvu dát, ktoré by užívateľ nikdy nevyhľadal v celku a taktiež vysokému IO costu. Stránkovanie bolo vytvorené v tvare:

```
DECLARE @PageNumber AS INT, @RowspPage AS INT
SET @PageNumber = 25
SET @RowspPage = 20
SELECT *
FROM League
ORDER BY leagueID
OFFSET ((@PageNumber - 1) * @RowspPage) ROWS
FETCH NEXT @RowspPage ROWS ONLY
GO
```

Vidíme, že stránkovanie bolo úspešné, v počte ms, MB aj IO coste.

Tabuľka TeamMatch

Kód	Operácia	Četnost	Očakávaná veľkosť výsledku	IO Cost	IO Cost 2	Čas operácie (ms)	Čas operácie 2 (ms)	Veľkosť výsledku (MB)	Veľkosť výsledku 2 (MB)
tm1	INSERT INTO *TeamMatch* VALUES (firstTeamID, secondTeamID, pitchID, firstTeamGoals, secondTeamGoals, postponed, date)	10	1	15	65	-	-	-	-
tm2	UPDATE *TeamMatch* SET firstTeamGoals = ftg, secondTeamGoals = stg, pitchID = pID, postponed = postponed, date = date WHERE teamMatchID = x	8	1	5	5	-	-	-	-
tm3	UPDATE *TeamMatch* SET postponed = y WHERE teamMatchID = x	5	1	3	3	-	-	-	-
tm4	SELECT * FROM *TeamMatch*	1	1200 000	5964	3	4080	66	39,68	0,0013
tm5	SELECT * FROM *TeamMatch* WHERE firstTeamID = x OR secondTeamID = x	4	250	783	783	-	-	-	-
tm6	SELECT * FROM *TeamMatch* WHERE teamMatchID = x	4	1	3	3	-	-	-	-
tm7	SELECT firstTeamID, secondTeamID, pitchID FROM *TeamMatch* WHERE teamMatchID = $t_m_id	6	1	3	3	-	-	-	-
tm8	SELECT firstTeamID, secondTeamID FROM *TeamMatch*	9	1	3	3	-	-	-	-

	WHERE teamMatchID = x								
tm9	UPDATE *TeamMatch* SET postponed = 1 WHERE (firstTeamID = $t_id OR secondTeamID = $t_id) AND date <= $new_date AND date >= $today	5	1	21	21	-	-	-	-
tm10	SELECT count(teamMatchID) FROM *TeamMatch* WHERE ((firstTeamID = $t_id OR secondTeamID = $t_id OR firstTeamID = $o_t_id OR secondTeamID = $o_t_id) AND date = $date	6	1	12	12	-	-	-	-
tm11	UPDATE *TeamMatch* SET date = $date WHERE teamMatchID = $t_m_id	6	1	3	3	-	-	-	-
tm12	SELECT * FROM *TeamMatch* WHERE pitchID = x	3	12	6046	39	-	-	-	-
tm13	SELECT * FROM *TeamMatch* WHERE firstTeamID = x	4	118	405	405	-	-	-	-

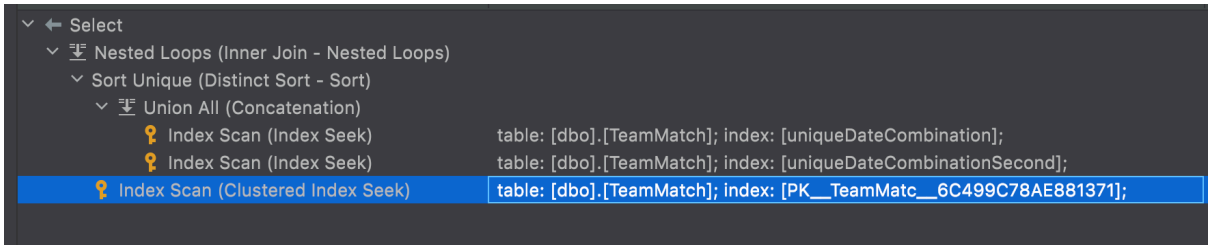
V tabuľke **TeamMatch** boli vytvorené indexy na atribúty *firstTeamIndex* a *pitchID*. Index *firstTeamIndex* bol vytvorený pre poukázanie zlého chovania systému – ukázané na dotaze **tm13** (popísané ďalej)**.** Pre dotaz **tm4** bolo vytvorené stránkovanie, z dôvodu prístupu k veľkému množstvu dát, ktoré by užívateľ nikdy nevyhľadal v celku a taktiež vysokému IO costu. Stránkovanie bolo vytvorené v tvare:

```
DECLARE @PageNumber AS INT, @RowspPage AS INT
SET @PageNumber = 30000
SET @RowspPage = 20
SELECT *
FROM TeamMatch
ORDER BY teamMatchID
OFFSET ((@PageNumber - 1) * @RowspPage) ROWS
FETCH NEXT @RowspPage ROWS ONLY
GO
```

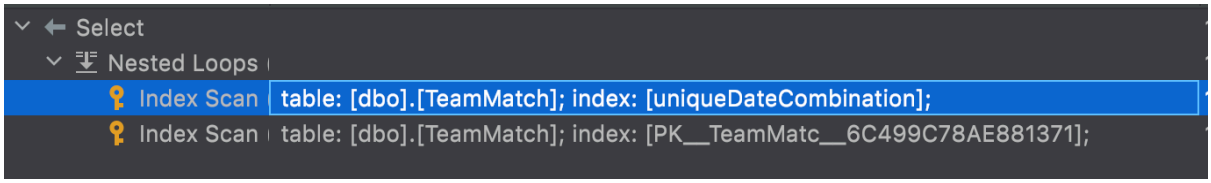
Vidíme, že stránkovanie bolo úspešné, v počte ms, MB aj IO coste.

Pre dotaz **tm5** som sa pokúsila vytvoriť dva indexy na atribút firstTeamID a secondTeamID, avšak obidva boli zbytočné – je to z dôvodu, že v dotaze je použité OR a teda index na atribúty v tomto prípade nehrá žiadnu rolu. Po vypísaní plánu je taktiež očividné, že program kontroluje dva indexy (index seek), ktoré vznikli pomocou UNIQUE kombinácie atribútov s dátumom a okrem toho robí **clustered index seek** na primárny kľúč. Toto všetko IO cost zväčšuje.

Priložený plán pre dotaz tm5:



Priložený plán pre dotaz tm13:



Na priloženom pláne pre dotaz **tm13** môžeme vidieť zlé chovanie sytému. Aj v prípade, že máme vytvorený nový index *firstTeamIndex*, tak ho systém nevyužije. Na toto chovanie som prišla pri testovaní dotazu **tm5.** Táto funkcionalita bola otestovaná aj v Microsoft SQL Management Studiu, kde sa systém choval rovnako.

Tabuľka Pitch

Kód	Operácia	Četnost	Očakávaná veľkosť výsledku	IO Cost	IO Cost 2	Čas operácie (ms)	Čas operácie 2 (ms)	Veľkosť výsledku (MB)	Veľkosť výsledku 2 (MB)
pi1	INSERT INTO *Pitch* VALUES (capacity, name)	3	1	2	2	-	-	-	-
pi2	UPDATE *Pitch* SET capacity = capacity, name = name WHERE pitchID = x	2	1	2	2	-	-	-	-
pi3	SELECT * FROM *Pitch*	1	100 010	371	2	601	60	1,9	0,0008
pi4	SELECT * FROM *Pitch* WHERE pitchID = x	3	1	2	2	-	-	-	-

V tabuľke **Pitch** nebol vytvorený žiadny index, pretože vidíme, že nie je potreba pre žiadny z dotazov. Pre dotaz **p3** bolo vytvorené stránkovanie, z dôvodu prístupu k veľkému množstvu dát, ktoré by užívateľ nikdy nevyhľadal v celku a taktiež vysokému IO costu. Stránkovanie bolo vytvorené v tvare:

```
DECLARE @PageNumber AS INT, @RowspPage AS INT
SET @PageNumber = 2500
SET @RowspPage = 20
SELECT *
FROM Pitch
ORDER BY pitchID
OFFSET ((@PageNumber - 1) * @RowspPage) ROWS
FETCH NEXT @RowspPage ROWS ONLY
GO
```

Vidíme, že stránkovanie bolo úspešné, v počte ms, MB aj IO coste.

Tabuľka PlayerTransferHistory

Kód	Operácia	Četnost	Očakávaná veľkosť výsledku	IO Cost	IO Cost 2	Čas operácie (ms)	Čas operácie 2 (ms)	Veľkosť výsledku (MB)	Veľkosť výsledku 2 (MB)
pth1	INSERT INTO *PlayerTransferHistory* VALUES (oldTeamID, newTeamID, playerID, date)	4	1	9	30	-	-	-	-
pth2	UPDATE *PlayerTransferHistory* SET oldTeamID = otID, newTeamID = ntID, playerID = pID, date = date WHERE playerTransferID = x	1	1	9	9	-	-	-	-
pth3	SELECT * FROM *PlayerTransferHistory*	1	100 000	349	2	406	59	2,1	0,001
pth4	SELECT * FROM *PlayerTransferHistory* WHERE playerID = x	3	1	349	4	-	-	-	-
pth5	SELECT * FROM *PlayerTransferHistory* WHERE oldTeamID = x OR newTeamID = x	3	21	349	54	-	-	-	-
pth6	SELECT * FROM *PlayerTransferHistory* WHERE playerTransferID = x	2	1	2	2	-	-	-	-

V tabuľke **PlayerTransferHistory** bol vytvorený index na atribút *playerID* pre dotaz **pth4** a indexy na atribúty *newTeamId* a *oldTeamID* pre dotaz **pth5.** Pre dotaz **pth3** bolo vytvorené stránkovanie, z dôvodu prístupu k veľkému množstvu dát, ktoré by užívateľ nikdy nevyhľadal v celku a taktiež vysokému IO costu. Stránkovanie bolo vytvorené v tvare:

```
DECLARE @PageNumber AS INT, @RowspPage AS INT
SET @PageNumber = 2500
SET @RowspPage = 20
SELECT *
FROM PlayerTransferHistory
ORDER BY playerTransferID
OFFSET ((@PageNumber - 1) * @RowspPage) ROWS
FETCH NEXT @RowspPage ROWS ONLY
GO
```

Vidíme, že stránkovanie bolo úspešné, v počte ms, MB aj IO coste.

Tabuľka Ticket

Kód	Operácia	Četnost	Očakávaná veľkosť výsledku	IO Cost	IO Cost 2	Čas operácie (ms)	Čas operácie 2 (ms)	Veľkosť výsledku (MB)	Veľkosť výsledku 2 (MB)
ti1	INSERT INTO *Ticket* VALUES (teamMatchID, firstName, lastName, price, storno, email)	7	1	6	18	-	-	-	-
ti2	UPDATE *Ticket* SET storno = 1 WHERE ticketID = x	3	1	3	3	-	-	-	-
ti3	SELECT * FROM *Ticket*	1	3 500 000	32419	4	52419	325	225	0,002
ti4	SELECT * FROM *Ticket* WHERE teamMatchID = x	3	5	32893	18	-	-	-	-
ti5	SELECT count(ticketID) from *Ticket* WHERE teamMatchID = x	7	1	32893	3	-	-	-	-

V tabuľke **Ticket** bol vytvorený index na atribút *teamMatchID*. Tento index pomohol zlepšiť operácie **ti4** a **ti5,** ktoré mali vysokú selektivitu, avšak obrovský IO cost. Ten sa podarilo rapídne znížiť. Pre dotaz **ti3** bolo vytvorené stránkovanie, z dôvodu prístupu k veľkému množstvu dát, ktoré by užívateľ nikdy nevyhľadal v celku a taktiež vysokému IO costu. Stránkovanie bolo vytvorené v tvare:

```
DECLARE @PageNumber AS INT, @RowspPage AS INT
SET @PageNumber = 75000
SET @RowspPage = 20
SELECT *
FROM Ticket
ORDER BY ticketID
OFFSET ((@PageNumber - 1) * @RowspPage) ROWS
FETCH NEXT @RowspPage ROWS ONLY
GO
```

Vidíme, že stránkovanie bolo úspešné, v počte ms, MB aj IO coste.

1.1. Suma dát

	IO Cost	IO Cost 2	Čas operácie (ms)	Čas operácie 2 (ms)	Veľkosť výsledku (MB)	Veľkosť výsledku 2 (MB)
SUMA	121 728	1937	64 526	643	283,17	0,0103

1.2. Legenda

Skupina 1 - červená – pridanie indexu
Skupina 2 – zelená – materializovaný pohľad
Skupina 3 – oranžová - stránkovanie
Skupina 4 – biela - ostatné

V návrhu som neuvažovala o iných možnostiach úpravy fyzického návrhu, pretože **indexy** a **stránkovanie** boli dostačujúce na vylepšenie v podstate všetkých dotazov.

2. Krížová validácia

Názov indexu	Typ	Operácie zlepšené	Operácie zhoršené	Veľkosť v blokoch
playerTeamID	B+ strom	**p4, p7, p9**	**p1**	349
playerTHplayerID	B+ strom	**pth4**	**pth1**	177
playerTHoldTeamID	B+ strom	**pth5**	**pth1**	176
playerTHnewTeamID	B+ strom	**pth5**	**pth1**	176
statisticTeamID	B+ strom	**s2, s5**	**s1**	176
statisticPlayerID	B+ strom	**s2, s4, s5**	**s1**	176
teamLeagueID	B+ strom	**t4, t7, t10**	**t1**	21
teamMatchPitch	B+ strom	**tm12**	**tm1**	2084
ticketTeamMatch	B+ strom	**ti3, ti4, ti5**	**ti1**	6073
firstTeamIndex	B+ strom	-	-	2084

3. Záver

V návrhu som neuvažovala o iných možnostiach úpravy fyzického návrhu, pretože **indexy** a **stránkovanie** boli dostačujúce na vylepšenie v podstate všetkých dotazov. Výsledné zlepšenie je obrovské, konkrétne:

- IO cost: ~62x
- Čas operácie(stránkovanie): ~100x
- Veľkost výsledku(stránkovanie): ~27500x