Milestone 8 Bewijs in PDF

Sam Rotthier - TIFL

Overzicht vergelijking:

Tabel info voor partitionering:

Query:

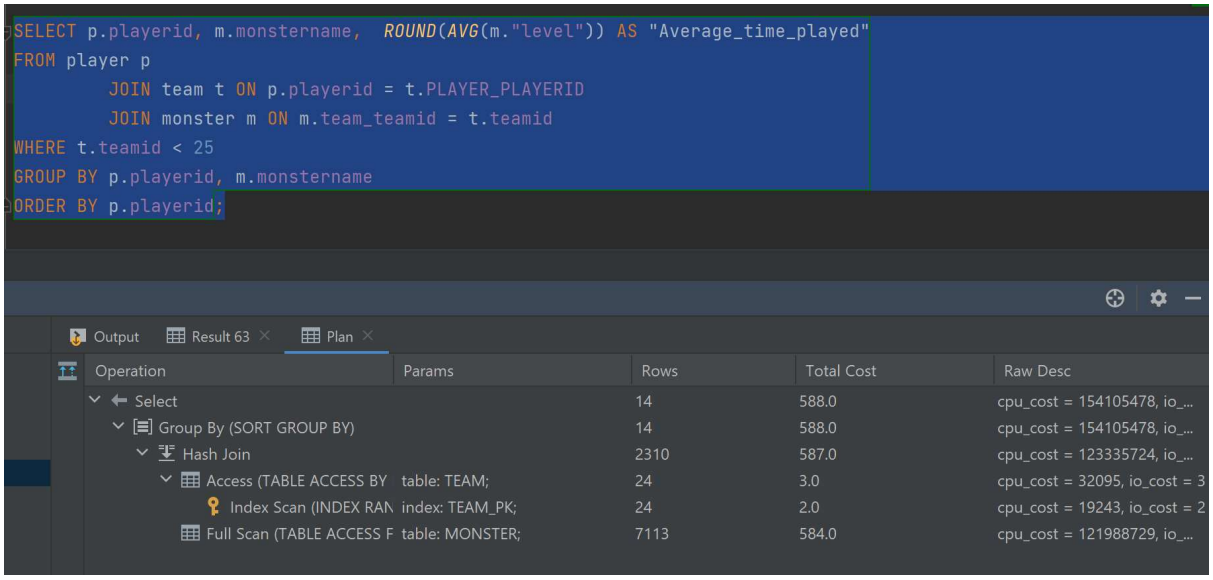
```
select segment_name,segment_type, sum(bytes/1024/1024) MB ,
(select   COUNT(*) FROM monster) as table_count
from dba_segments
where segment_name=  'MONSTER'
group by segment_name,segment_type;
```

	SEGMENT_NAME	SEGMENT_TYPE	MB	TABLE_COUNT
1	MONSTER	TABLE	17.0625	400000

Query:

```
SELECT p.playerid, m.monstername,  ROUND(AVG(m."level")) AS "Average_time_played"
FROM player p
        JOIN team t ON p.playerid = t.PLAYER_PLAYERID
        JOIN monster m ON m.team_teamid = t.teamid
WHERE t.teamid < 25
GROUP BY p.playerid, m.monstername
ORDER BY p.playerid;
```

Explain plan



NA partitionering:

Partitie script + uitleg partitie sleutel

```
DROP TABLE monster CASCADE CONSTRAINTS PURGE;
CREATE TABLE monster (
                monsterid   INTEGER GENERATED ALWAYS AS IDENTITY
                   CONSTRAINT software_pk PRIMARY KEY,
                monstername VARCHAR2(500) NOT NULL,
                health      INTEGER NOT NULL,
                "level"     INTEGER,
                canevolve   CHAR(1),
                team_teamid INTEGER NOT NULL
)
   PARTITION BY RANGE (team_teamid)
   INTERVAL (50)
(
   partition team_50 VALUES LESS THAN (50)
);
ALTER TABLE monster
   ADD CONSTRAINT monster_team_fk FOREIGN KEY (team_teamid)
   REFERENCES team ( teamid );
```

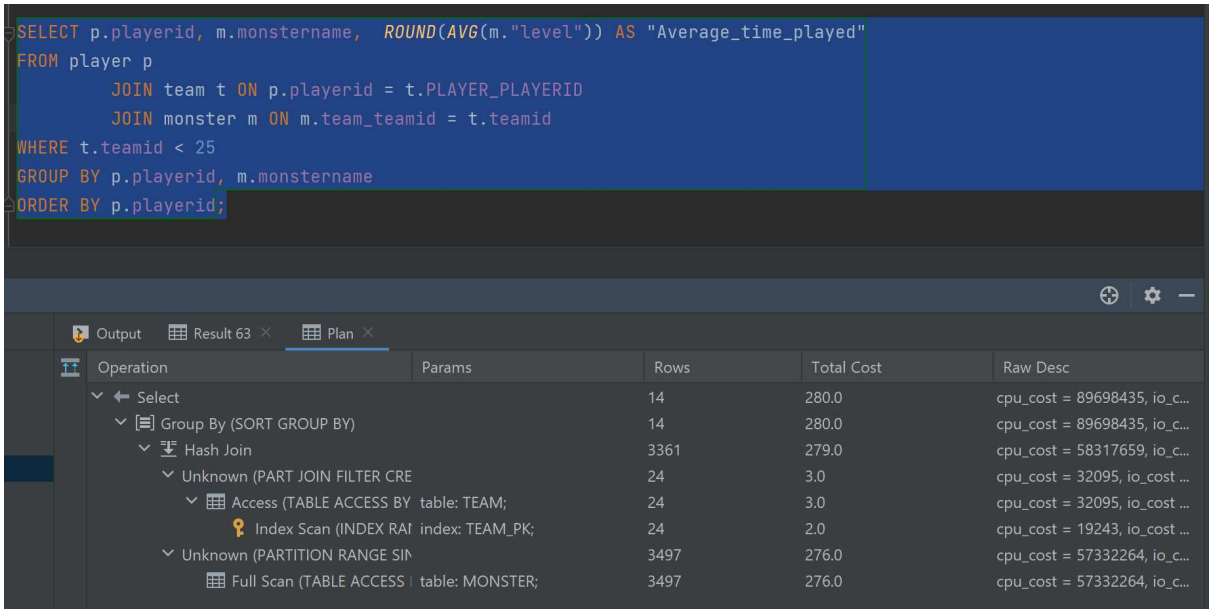
Tabel info NA partitionering:

Query:

```
select segment_name,segment_type, sum(bytes/1024/1024) MB ,
(select   COUNT(*) FROM monster) as table_count
from dba_segments
where segment_name=  'MONSTER'
group by segment_name,segment_type;
```

	SEGMENT_NAME	SEGMENT_TYPE	MB	TABLE_COUNT
1	MONSTER	TABLE	0.0625	400000
2	MONSTER	TABLE PARTITION	72	400000

Explain plan na partitionering



```
SELECT p.playerid, m.monstername,  ROUND(AVG(m."level")) AS "Average_time_played"
FROM player p
        JOIN team t ON p.playerid = t.PLAYER_PLAYERID
        JOIN monster m ON m.team_teamid = t.teamid
WHERE t.teamid < 25
GROUP BY p.playerid, m.monstername
ORDER BY p.playerid;
```

Output | Result 63 | Plan

Operation	Params	Rows	Total Cost	Raw Desc
Select		14	280.0	cpu_cost = 89698435, io_c...
Group By (SORT GROUP BY)		14	280.0	cpu_cost = 89698435, io_c...
Hash Join		3361	279.0	cpu_cost = 58317659, io_c...
Unknown (PART JOIN FILTER CRE		24	3.0	cpu_cost = 32095, io_cost ...
Access (TABLE ACCESS BY	table: TEAM;	24	3.0	cpu_cost = 32095, io_cost ...
Index Scan (INDEX RAI	index: TEAM_PK;	24	2.0	cpu_cost = 19243, io_cost ...
Unknown (PARTITION RANGE SIN		3497	276.0	cpu_cost = 57332264, io_c...
Full Scan (TABLE ACCESS	table: MONSTER;	3497	276.0	cpu_cost = 57332264, io_c...

Blooper:

Voor:

```
SELECT p.playerid, p.name, ROUND(AVG(m."level")) AS "Average_monster_level"
FROM player p
        JOIN team t ON p.playerid = t.PLAYER_PLAYERID
        JOIN monster m ON m.team_teamid = t.teamid
WHERE m."level" > 25
GROUP BY p.playerid, p.name;
```

Output | Result 63 | Plan

Operation	Params	Rows	Total Cost	Raw Desc
Select		20	599.0	cpu_cost = 426815600, i...
Merge Join		20	599.0	cpu_cost = 426815600, i...
Index Scan (TABLE ACCESS BY INDI	table: PLAYER;	20	2.0	cpu_cost = 22643, io_cost...
Full Index Scan (INDEX FULL SC.	index: PLAYER_PK;	20	1.0	cpu_cost = 11121, io_cost...
Sort (SORT JOIN)		20	597.0	cpu_cost = 426792957, i...
Access (VIEW)		20	596.0	cpu_cost = 397182221, i...
Group By (HASH GROUP BY)		20	596.0	cpu_cost = 397182221, i...
Hash Join		269799	588.0	cpu_cost = 148268422, i...
Full Scan (TABLE ACC	table: TEAM;	800	3.0	cpu_cost = 203607, io_co...
Full Scan (TABLE ACC	table: MONSTER;	269799	584.0	cpu_cost = 120364915, i...

Na:

```
SELECT p.playerid, p.name, ROUND(AVG(m."level")) AS "Average_monster_level"
FROM player p
        JOIN team t ON p.playerid = t.PLAYER_PLAYERID
        JOIN monster m ON m.team_teamid = t.teamid
WHERE m."level" > 25
GROUP BY p.playerid, p.name;
```

Output | Result 63 | Plan

Operation	Params	Rows	Total Cost	Raw Desc
Select		20	917.0	cpu_cost = 2755834168,...
Merge Join		20	917.0	cpu_cost = 2755834168,...
Index Scan (TABLE ACCESS BY INDEX	table: PLAYER;	20	2.0	cpu_cost = 22643, io_cos...
Full Index Scan (INDEX FULL SCAN	index: PLAYER_PK;	20	1.0	cpu_cost = 11121, io_cos...
Sort (SORT JOIN)		20	915.0	cpu_cost = 2755811526,...
Access (VIEW)		20	914.0	cpu_cost = 2726200789,...
Group By (HASH GROUP BY)		20	914.0	cpu_cost = 2726200789,...
Hash Join		230209	907.0	cpu_cost = 2511842411,...
Full Scan (TABLE ACCES	table: TEAM;	800	3.0	cpu_cost = 203607, io_c...
Unknown (PARTITION RANG		230209	903.0	cpu_cost = 2487897904,...
Full Scan (TABLE ACC	table: MONSTER;	230209	903.0	cpu_cost = 2487897904,...

Conclusie:

In de blooper kunnen we zien dat wanneer je een index op een verkeerd attribuut zet (of een verkeerde filter toepast) dat dit tegenovergesteld werkt en de query langer laat duren.
Dit komt doordat het programma nu over meerdere files gaat moeten zoeken in plaats van (bij een goede partitionering) minder.
Door de blooper zie je duidelijk hoe de theorie de voor en nadelen van een partitie bespreekt.

Bij een goede parititie (en filtering) gaat het syteem net over minder files moeten zoeken en hierdoor veel tijd winnen.