Bewijs Milestone 8

Arthur Linsen – INF201B

Overzicht vergelijking

Tabel info voor partitionering

SEGMENT_NAME	SEGMENT_TYPE	MB	TABLE_COUNT
EXERCISES	TABLE	40	560000

Query

```
-- Ik wil een overzicht zien voor een bepaalde fitness van het gemiddelde maxgewicht per machine
SELECT f.NAME, m.NAME, AVG(e.MAXWEIGHT)
FROM FITNESS f
         JOIN MACHINES m on f.FITNESSID = m.FITNESS_FITNESSID
         JOIN EXERCISES e on m.MACHINEID = e.MACHINE_MACHINEID
WHERE m.MACHINEID = 756
-- WHERE m.NAME = 'Machine36'
GROUP BY f.NAME, m.NAME
ORDER BY m.NAME;
```

Ik moest de ID gebruiken omdat wanneer ik een naam gebruikte het explain via een hash join te werk ging en niet via een nested loop op men foreign key.

Explain plan

Output | Plan

Operation	Params	Rows	Total Cost	Raw Desc
Select		1	1372.0	cpu_cost = 182100153, io_cost = 1367
Group By (HASH GROUP BY)		1	1372.0	cpu_cost = 182100153, io_cost = 1367
Nested Loops		799	1372.0	cpu_cost = 182100153, io_cost = 1367
Nested Loops		1	3.0	cpu_cost = 23884, io_cost = 3
Index Scan (TABLE ACCESS BY INDEX ROWII	table: MACHINES;	1	2.0	cpu_cost = 15543, io_cost = 2
Unique Index Scan (INDEX UNIQUE SCAI	index: MACHINES_PK;	1	1.0	cpu_cost = 8171, io_cost = 1
Index Scan (TABLE ACCESS BY INDEX ROWII	table: FITNESS;	1	1.0	cpu_cost = 8341, io_cost = 1
Unique Index Scan (INDEX UNIQUE SCAI	index: FITNESS_PK;	1	0.0	cpu_cost = 1050, io_cost = 0
Access (TABLE ACCESS STORAGE FULL)	table: EXERCISES;	799	1369.0	cpu_cost = 182076269, io_cost = 1364

Na partitionering

Partitie script + uitleg partitie sleutel

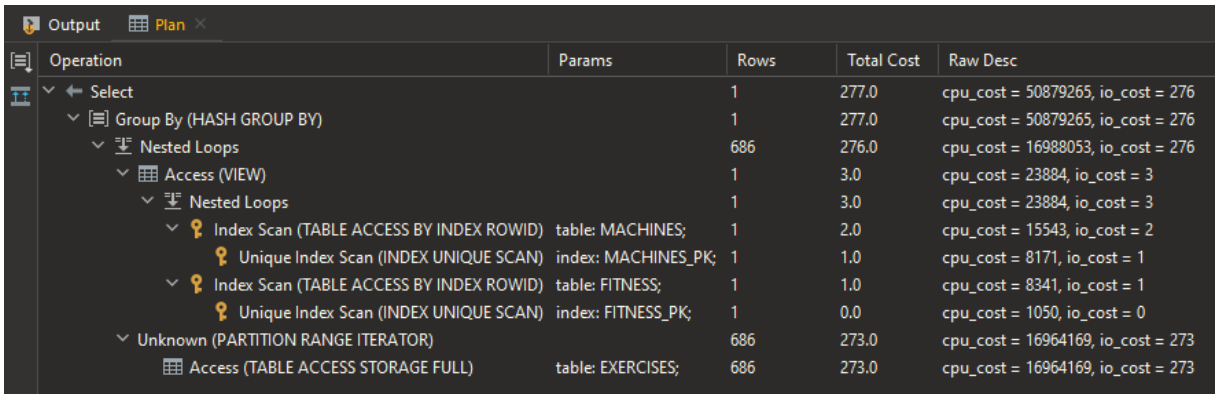
```
CREATE TABLE exercises
(
    exerciseid                NUMBER GENERATED ALWAYS AS IDENTITY NOT NULL
        CONSTRAINT exercises_pk PRIMARY KEY,
    name                      VARCHAR2(30)                        NOT NULL,
    maxweight                 NUMBER                              NOT NULL,
    instruction               VARCHAR2(200)                       NOT NULL,
    machine_machineid         NUMBER                              NOT NULL,
    difficulty_difficultyid   NUMBER                              NOT NULL,
    musclegroup_muclsegroupid NUMBER                              NOT NULL
)
    PARTITION BY RANGE (machine_machineid)
(
    PARTITION machine_0001 VALUES LESS THAN (50),
    PARTITION machine_0002 VALUES LESS THAN (100),
    PARTITION machine_0003 VALUES LESS THAN (150),
    PARTITION machine_0004 VALUES LESS THAN (200),
    PARTITION machine_0005 VALUES LESS THAN (250),
    PARTITION machine_0006 VALUES LESS THAN (300),
    PARTITION machine_0007 VALUES LESS THAN (350),
    PARTITION machine_0008 VALUES LESS THAN (400),
    PARTITION machine_0009 VALUES LESS THAN (450),
    PARTITION machine_0010 VALUES LESS THAN (500),
    PARTITION machine_0011 VALUES LESS THAN (550),
    PARTITION machine_0012 VALUES LESS THAN (600),
    PARTITION machine_0013 VALUES LESS THAN (650),
    PARTITION machine_0014 VALUES LESS THAN (700),
    PARTITION machine_0015 VALUES LESS THAN (750),
    PARTITION machine_0016 VALUES LESS THAN (maxvalue)
);
```

De partitie sleutel die hier gebruikt wordt in tabel Z is de foreign key machine_machineid. Van deze sleutel zijn er 800 verschillende . Daardoor wordt machine_machineid opgesplitst in 10 parities van elk 50 id’s.

Tabel info NA partitionering

SEGMENT_NAME	SEGMENT_TYPE	MB	TABLE_COUNT
EXERCISES	TABLE PARTITION	128	560000

Explain plan NA partitionering



Output | Plan

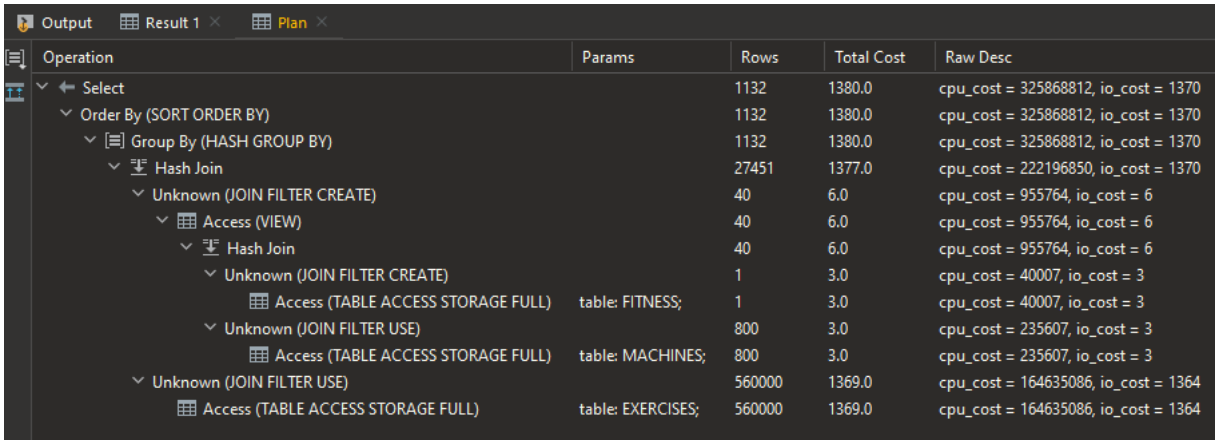
Operation	Params	Rows	Total Cost	Raw Desc
Select		1	277.0	cpu_cost = 50879265, io_cost = 276
Group By (HASH GROUP BY)		1	277.0	cpu_cost = 50879265, io_cost = 276
Nested Loops		686	276.0	cpu_cost = 16988053, io_cost = 276
Access (VIEW)		1	3.0	cpu_cost = 23884, io_cost = 3
Nested Loops		1	3.0	cpu_cost = 23884, io_cost = 3
Index Scan (TABLE ACCESS BY INDEX ROWID)	table: MACHINES;	1	2.0	cpu_cost = 15543, io_cost = 2
Unique Index Scan (INDEX UNIQUE SCAN)	index: MACHINES_PK;	1	1.0	cpu_cost = 8171, io_cost = 1
Index Scan (TABLE ACCESS BY INDEX ROWID)	table: FITNESS;	1	1.0	cpu_cost = 8341, io_cost = 1
Unique Index Scan (INDEX UNIQUE SCAN)	index: FITNESS_PK;	1	0.0	cpu_cost = 1050, io_cost = 0
Unknown (PARTITION RANGE ITERATOR)		686	273.0	cpu_cost = 16964169, io_cost = 273
Access (TABLE ACCESS STORAGE FULL)	table: EXERCISES;	686	273.0	cpu_cost = 16964169, io_cost = 273

Conclusie

Elke rij wordt toegewezen aan 1 van de 10 partities. Dit wordt gedaan omdat grote tabellen moeilijker beheersbaar zijn en deze dan worden opgesplitst in kleinere en beter beheersbare stukken die partities noemen. De partitiesleutel is machine_machineid. Er wordt gebruik gemaakt van range partitionering de 800 verschillende id's worden verdeelt over 10 partities elk bevat 50 id's.

Na parititionering is de CPU kost kleiner geworden. De tijd voor het genereren van een 560 000 rijen is van 12 seconden voor paritionering naar 43 seconden na partitionering gegaan. De total cost is van 1369.0 gegaan naar 273.0 .

--



Output | Result 1 | Plan

Operation	Params	Rows	Total Cost	Raw Desc
Select		1132	1380.0	cpu_cost = 325868812, io_cost = 1370
Order By (SORT ORDER BY)		1132	1380.0	cpu_cost = 325868812, io_cost = 1370
Group By (HASH GROUP BY)		1132	1380.0	cpu_cost = 325868812, io_cost = 1370
Hash Join		27451	1377.0	cpu_cost = 222196850, io_cost = 1370
Unknown (JOIN FILTER CREATE)		40	6.0	cpu_cost = 955764, io_cost = 6
Access (VIEW)		40	6.0	cpu_cost = 955764, io_cost = 6
Hash Join		40	6.0	cpu_cost = 955764, io_cost = 6
Unknown (JOIN FILTER CREATE)		1	3.0	cpu_cost = 40007, io_cost = 3
Access (TABLE ACCESS STORAGE FULL)	table: FITNESS;	1	3.0	cpu_cost = 40007, io_cost = 3
Unknown (JOIN FILTER USE)		800	3.0	cpu_cost = 235607, io_cost = 3
Access (TABLE ACCESS STORAGE FULL)	table: MACHINES;	800	3.0	cpu_cost = 235607, io_cost = 3
Unknown (JOIN FILTER USE)		560000	1369.0	cpu_cost = 164635086, io_cost = 1364
Access (TABLE ACCESS STORAGE FULL)	table: EXERCISES;	560000	1369.0	cpu_cost = 164635086, io_cost = 1364