Milestone 8 Bewijs in PDF

Maxim Derboven - INF203

Overzicht vergelijking:

Tabel info voor partitionering:

	SEGMENT_NAME	SEGMENT_TYPE	MB	TABLE_COUNT
1	PERFORMANCES	TABLE	19	502505

Query:

Test 1

```
-- Gemiddelde dagen tussen start en einde van film performances (uit 2019) (in dagen) per hall met minder dan 4 stoelen
SELECT T.THEATHER_ID, T.NAME,H.HALL_ID, ROUND(AVG(CAST(P.ENDTIME as DATE)-CAST(P.STARTTIME AS DATE))) AS "Gemiddelde dagen per performance"
FROM THEATHERS T
        JOIN HALLS H on H.THEATHER_ID = T.THEATHER_ID
        JOIN PERFORMANCES P on P.HALL_ID = H.HALL_ID
WHERE H.SEAT_AMOUNT < 4 AND P.STARTTIME BETWEEN TO_DATE('01-01-2019','DD-MM-YYYY') AND TO_DATE('31-12-2019','DD-MM-YYYY')
group by T.THEATHER_ID, T.NAME, H.SEAT_AMOUNT, H.HALL_ID
ORDER BY T.THEATHER_ID DESC;
```

Test 2

```
SELECT T.THEATHER_ID,T.NAME, ROUND(AVG(CAST(P.ENDTIME as DATE)-CAST(P.STARTTIME AS DATE))) AS "Gemiddelde dagen per performance"
FROM THEATHERS T
        JOIN HALLS H on H.THEATHER_ID = T.THEATHER_ID
        JOIN PERFORMANCES P on P.HALL_ID = H.HALL_ID
WHERE P.STARTTIME BETWEEN TO_DATE('01-11-2019 00:00','DD-MM-YYYY HH24:MI') AND TO_DATE('01-11-2019 23:59','DD-MM-YYYY HH24:MI')
GROUP BY T.THEATHER_ID, T.NAME ;
```

Explain plan

Test 1

Output | Gemiddelde dagen tus... minder dan 5 stoelen | Plan

Operation	Params	Rows	Total Cost	Raw Desc
Select		77	464.0	cpu_cost = 53325643, io_cost = 462
Group By (SORT GROUP BY)		77	464.0	cpu_cost = 53325643, io_cost = 462
Nested Loops		77	463.0	cpu_cost = 23697061, io_cost = 462
Nested Loops		77	463.0	cpu_cost = 23697061, io_cost = 462
Hash Join		19	6.0	cpu_cost = 903434, io_cost = 6
Full Scan (TABLE ACCESS FULL	table: HALLS;	19	3.0	cpu_cost = 258227, io_cost = 3
Full Scan (TABLE ACCESS FULL	table: THEATHERS;	25	3.0	cpu_cost = 39857, io_cost = 3
Index Scan (INDEX RANGE SCAN)	index: UNIQUE_PERFORMANCE;	4	19.0	cpu_cost = 1162907, io_cost = 19
Index Scan (TABLE ACCESS BY INDEX	table: PERFORMANCES;	4	24.0	cpu_cost = 1199665, io_cost = 24

Test 2

Operation	Params	Rows	Total Cost	Raw Desc
Select		11	2.0	cpu_cost = 29643598, io_cost = 1
Group By (HASH GROUP BY)		11	2.0	cpu_cost = 29643598, io_cost = 1
Nested Loops		11	1.0	cpu_cost = 35041, io_cost = 1
Nested Loops		11	1.0	cpu_cost = 35041, io_cost = 1
Nested Loops		11	1.0	cpu_cost = 21621, io_cost = 1
Access (TABLE ACCESS BY INDEX ROWID BATCHED)	table: PERFORMANCES;	11	1.0	cpu_cost = 7321, io_cost = 1
Index Scan (INDEX RANGE SCAN)	index: UNIQUE_PERFORMANCE;	11	1.0	cpu_cost = 7321, io_cost = 1
Index Scan (TABLE ACCESS BY INDEX ROWID)	table: HALLS;	1	0.0	cpu_cost = 1300, io_cost = 0
Unique Index Scan (INDEX UNIQUE SCAN)	index: HALL_PK;	1	0.0	cpu_cost = 1050, io_cost = 0
Unique Index Scan (INDEX UNIQUE SCAN)	index: THEATHER_PK;	1	0.0	cpu_cost = 1050, io_cost = 0
Index Scan (TABLE ACCESS BY INDEX ROWID)	table: THEATHERS;	1	0.0	cpu_cost = 1220, io_cost = 0

NA partitionering:

Partitie script + uitleg partitie sleutel

Test 1
Ik partitioneer de performances elk jaar op hun start tijd (‘starttime’). De eerste performance vondt plaats op 2018/01/01

```
CREATE TABLE PERFORMANCES
(
    performance_id INTEGER GENERATED ALWAYS AS IDENTITY CONSTRAINT fk_performance PRIMARY KEY,
    movie_id       INTEGER NOT NULL CONSTRAINT fk_movie_performance REFERENCES MOVIES(MOVIE_ID) ON DELETE CASCADE,
    hall_id        INTEGER NOT NULL CONSTRAINT fk_hall_performance REFERENCES HALLS(HALL_ID) ON DELETE CASCADE,
    starttime      TIMESTAMP NOT NULL,
    endtime        TIMESTAMP NOT NULL
)
PARTITION BY RANGE (starttime)
    INTERVAL (NUMTOYMINTERVAL(1, 'YEAR'))
(
    PARTITION p0 VALUES LESS THAN (TO_DATE('2019/01/01', 'YYYY/MM/DD')),
    PARTITION p1 VALUES LESS THAN (TO_DATE('2020/01/01', 'YYYY/MM/DD'))
);
```

Test 2
Ik partitioneer de performances elke 2 maanden op hun start tijd (‘starttime’). De eerste performance vondt plaats op 2018/01/01

```
DROP TABLE PERFORMANCES CASCADE CONSTRAINTS PURGE;
CREATE TABLE PERFORMANCES
(
    performance_id INTEGER GENERATED ALWAYS AS IDENTITY CONSTRAINT fk_performance PRIMARY KEY,
    movie_id       INTEGER NOT NULL CONSTRAINT fk_movie_performance REFERENCES MOVIES(MOVIE_ID) ON DELETE CASCADE,
    hall_id        INTEGER NOT NULL CONSTRAINT fk_hall_performance REFERENCES HALLS(HALL_ID) ON DELETE CASCADE,
    starttime      TIMESTAMP NOT NULL,
    endtime        TIMESTAMP NOT NULL
)
PARTITION BY RANGE (starttime)
    INTERVAL (NUMTOYMINTERVAL(2, 'MONTH'))
(
    PARTITION p0 VALUES LESS THAN (TO_DATE('2018/03/01', 'YYYY/MM/DD')),
    PARTITION p1 VALUES LESS THAN (TO_DATE('2018/05/01', 'YYYY/MM/DD'))
);
```

Tabel info NA partitionering:

Test 1

	SEGMENT_NAME	SEGMENT_TYPE	MB	TABLE_COUNT
1	PERFORMANCES	TABLE PARTITION	64	502505

Test 2

	SEGMENT_NAME	SEGMENT_TYPE	MB	TABLE_COUNT
1	PERFORMANCES	TABLE PARTITION	216	502505

Query: → moet dezelfde zijn

Explain plan na partitionering

Test 1

Operation	Params	Rows	Total Cost	Raw Desc
Select		105	281.0	cpu_cost = 69043679, io_cost = 279
Group By (SORT GROUP BY)		105	281.0	cpu_cost = 69043679, io_cost = 279
Hash Join		105	280.0	cpu_cost = 39405074, io_cost = 279
Merge Join		26	6.0	cpu_cost = 29894628, io_cost = 5
Index Scan (TABLE ACCESS BY I	table: THEATHERS;	25	2.0	cpu_cost = 23493, io_cost = 2
Full Index Scan (INDEX FULL	index: THEATHER_PK;	25	1.0	cpu_cost = 12121, io_cost = 1
Sort (SORT JOIN)		26	4.0	cpu_cost = 29871135, io_cost = 3
Full Scan (TABLE ACCESS FU	table: HALLS;	26	3.0	cpu_cost = 258787, io_cost = 3
Unknown (PARTITION RANGE SINGLE)		4046	274.0	cpu_cost = 8501946, io_cost = 274
Full Scan (TABLE ACCESS FULL)	table: PERFORMANCES;	4046	274.0	cpu_cost = 8501946, io_cost = 274

Test 2

Output | Get table size | Plan

Operation	Params	Rows	Total Cost	Raw Desc
Select		9	281.0	cpu_cost = 67566611, io_cost = 279
Merge Join		9	281.0	cpu_cost = 67566611, io_cost = 279
Index Scan (TABLE ACCESS BY INDEX ROWID)	table: THEATHERS;	25	2.0	cpu_cost = 24743, io_cost = 2
Full Index Scan (INDEX FULL SCAN)	index: THEATHER_PK;	25	1.0	cpu_cost = 12121, io_cost = 1
Sort (SORT JOIN)		9	279.0	cpu_cost = 67541868, io_cost = 277
Access (VIEW)		9	278.0	cpu_cost = 37933741, io_cost = 277
Group By (HASH GROUP BY)		9	278.0	cpu_cost = 37933741, io_cost = 277
Hash Join		9	277.0	cpu_cost = 8325614, io_cost = 277
Unknown (PARTITION RANGE SINGLE)		9	274.0	cpu_cost = 7336907, io_cost = 274
Full Scan (TABLE ACCESS FULL)	table: PERFORMANCES;	9	274.0	cpu_cost = 7336907, io_cost = 274
Full Scan (TABLE ACCESS FULL)	table: HALLS;	1005	3.0	cpu_cost = 286857, io_cost = 3

Conclusie:

De costs dalen omdat hij sneller doorheen de data kan door de bepaalde partitie grenzen. De tijd zal daarom ook afnemen. Hoe meer partities hoe groter de database wordt.
Er wordt bij mij in eerste instantie sowieso gezocht op auto indexes (pk’s) dus veel verschil is er niet.