MILESTONE 8 – Bewijs pdf

Justin van Leuvenum – INF 203A

Overzicht vergelijking :

Tabel ter info voor partitionering:

	SEGMENT_NAME	SEGMENT_TYPE	MB	TABLE_COUNT
1	ORDERLIJN	TABLE	47	1040005

Query:

```
-- Ik wil een overzicht van de klanten waarbij we de prijs van de orderlijnen zien die tussen 650 en 850 euro zijn
SELECT W.WINKEL_NAAM, K.NAAM, O.AANTAL, O.CONSOLE_CONSOLENAAM, o.prijs
FROM WINKEL W
        JOIN KLANTORDER KO on W.WINKEL_ID = KO.WINKEL_WINKEL_ID
        JOIN KLANT K on K.KLANTID = KO.KLANT_KLANTID
        JOIN ORDERLIJN O on KO.ORDERID = O.KLANTORDER_ORDERID and
                            KO.WINKEL_WINKEL_ID = O.KLANTORDER_WINKEL_ID and
                            KO.KLANT_KLANTID = O.KLANTORDER_KLANTID
WHERE o.prijs BETWEEN 550 AND 580
GROUP BY WINKEL_NAAM, NAAM, AANTAL, o.PRIJS, CONSOLE_CONSOLENAAM
```

Explain plan:

Operation	Params	Rows	Total Cost	Raw Desc
Select		148	1624.0	cpu_cost = 391456779, io_cost = 1611
Group By (HASH GROUP BY)		148	1624.0	cpu_cost = 391456779, io_cost = 1611
Hash Join		17725	1623.0	cpu_cost = 350579008, io_cost = 1611
Hash Join		1045	9.0	cpu_cost = 30998557, io_cost = 8
Full Scan (TABLE ACCESS FULL)	table: WINKEL;	26	3.0	cpu_cost = 40027, io_cost = 3
Merge Join		1045	6.0	cpu_cost = 30250129, io_cost = 5
Index Scan (TABLE ACCESS BY INDE	table: KLANT;	8	2.0	cpu_cost = 17203, io_cost = 2
Full Index Scan (INDEX FULL SCA	index: KLANT_PK;	8	1.0	cpu_cost = 8721, io_cost = 1
Sort (SORT JOIN)		1045	4.0	cpu_cost = 30232927, io_cost = 3
Full Index Scan (INDEX FAST FUL	index: KLANTORDER_PK;	1045	3.0	cpu_cost = 153886, io_cost = 3
Full Scan (TABLE ACCESS FULL)	table: ORDERLIJN;	215345	1613.0	cpu_cost = 297289201, io_cost = 1603

NA partitionering:

```
CREATE TABLE orderlijn
(
    orderlijnid          NUMBER GENERATED ALWAYS AS IDENTITY,
    console_consolenaam  VARCHAR2(50),
    aantal               NUMBER,
    prijs                NUMBER,
    klantorder_winkel_id NUMBER NOT NULL,
    klantorder_klantid   NUMBER NOT NULL,
    klantorder_orderid   NUMBER NOT NULL,
    constraint ch_aantal CHECK ( aantal < 4 )


)
    PARTITION BY RANGE (prijs)
(
    -- interval werkt niet
    PARTITION prijs_100 VALUES LESS THAN (100),
    PARTITION prijs_200 VALUES LESS THAN (200),
    PARTITION prijs_300 VALUES LESS THAN (300),
    PARTITION prijs_400 VALUES LESS THAN (400),
    PARTITION prijs_500 VALUES LESS THAN (500),
    PARTITION prijs_600 VALUES LESS THAN (600),
    PARTITION prijs_700 VALUES LESS THAN (700),
    PARTITION prijs_800 VALUES LESS THAN (800),
    PARTITION prijs_900 VALUES LESS THAN (MAXVALUE)
);
```

Ik heb gebruik gemaakt van de range partitioning. Dit leek me logisch sinds de prijs van een orderlijn alleen tussen 50 en 900 euro kan liggen. Ik splits mijn data op in 9 delen van 100. Waardoor ik 9 partitites zal hebben.

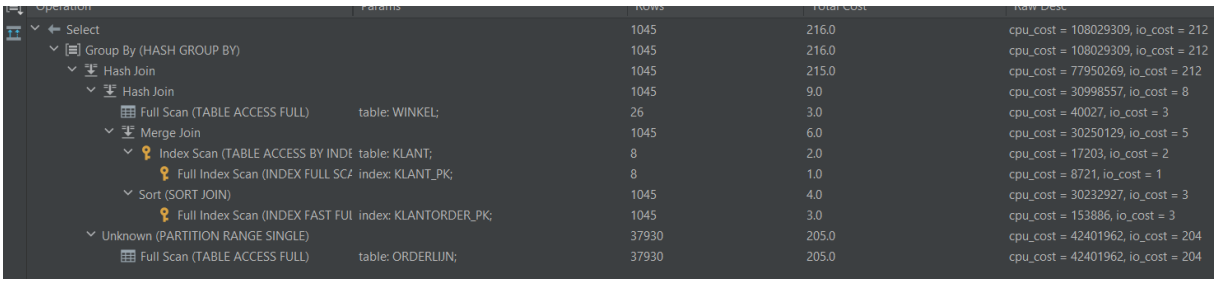
Tabel info NA partitionering:

	SEGMENT_NAME	SEGMENT_TYPE	MB	TABLE_COUNT
1	ORDERLIJN	TABLE PARTITION	51	1040005

Query:

```
-- Ik wil een overzicht van de klanten waarbij we de prijs van de orderlijnen zien die tussen 650 en 850 euro zijn
SELECT W.WINKEL_NAAM, K.NAAM, O.AANTAL, O.CONSOLE_CONSOLENAAM, o.prijs
FROM WINKEL W
         JOIN KLANTORDER KO on W.WINKEL_ID = KO.WINKEL_WINKEL_ID
         JOIN KLANT K on K.KLANTID = KO.KLANT_KLANTID
         JOIN ORDERLIJN O on KO.ORDERID = O.KLANTORDER_ORDERID and
                             KO.WINKEL_WINKEL_ID = O.KLANTORDER_WINKEL_ID and
                             KO.KLANT_KLANTID = O.KLANTORDER_KLANTID
WHERE o.prijs BETWEEN 550 AND 580
GROUP BY WINKEL_NAAM, NAAM, AANTAL, o.PRIJS, CONSOLE_CONSOLENAAM
```

Explain plan na partitionering



Operation	Params	Rows	Total Cost	Raw Desc
Select		1045	216.0	cpu_cost = 108029309, io_cost = 212
Group By (HASH GROUP BY)		1045	216.0	cpu_cost = 108029309, io_cost = 212
Hash Join		1045	215.0	cpu_cost = 77950269, io_cost = 212
Hash Join		1045	9.0	cpu_cost = 30998557, io_cost = 8
Full Scan (TABLE ACCESS FULL)	table: WINKEL;	26	3.0	cpu_cost = 40027, io_cost = 3
Merge Join		1045	6.0	cpu_cost = 30250129, io_cost = 5
Index Scan (TABLE ACCESS BY INDE	table: KLANT;	8	2.0	cpu_cost = 17203, io_cost = 2
Full Index Scan (INDEX FULL SC/	index: KLANT_PK;	8	1.0	cpu_cost = 8721, io_cost = 1
Sort (SORT JOIN)		1045	4.0	cpu_cost = 30232927, io_cost = 3
Full Index Scan (INDEX FAST FUI	index: KLANTORDER_PK;	1045	3.0	cpu_cost = 153886, io_cost = 3
Unknown (PARTITION RANGE SINGLE)		37930	205.0	cpu_cost = 42401962, io_cost = 204
Full Scan (TABLE ACCESS FULL)	table: ORDERLIJN;	37930	205.0	cpu_cost = 42401962, io_cost = 204

Conclusie:

De totale cost om die query uit te voeren is van 1624 (zonder partities) naar 216 (met partities).

De cpu kost om die query uit te voeren is van 1611 (zonder partities) naar 212 (met partities).

Na het toevoegen van de range partities is de performantie van dezelfde query ongeveer 9 keer versnelt en efficiënter geworden. Dit is ook logisch. Hoe meer partitites je hebt, hoe sneller je querie zal zijn.