Advanced Database Project

Final Phase

Team 16

Esraa Mamdouh

Khaled Sabry

Mohamed Emad

Roba Gamal

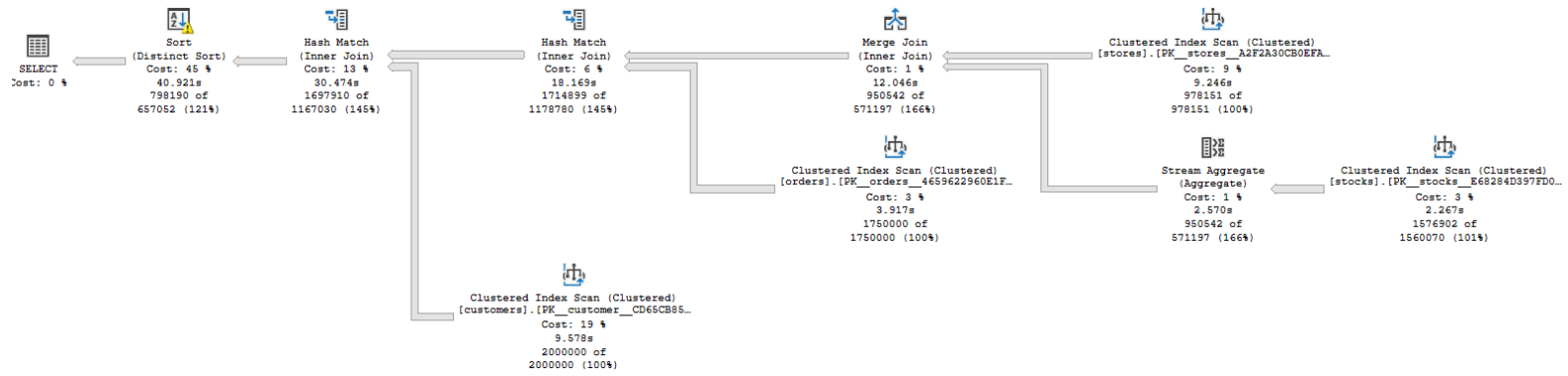
Submitted to:

E. Mostafa Mahmoud

SQL Query #1

```sql
SELECT DISTINCT
       c.first_name,
       c.last_name
FROM
       sales.orders AS o,
       sales.customers AS c
WHERE
       o.customer_id = c.customer_id
       AND o.store_id IN (
       SELECT
              sto.store_id
       FROM
              sales.stores AS sto
WHERE
          sto.store_id IN ( SELECT stc.store_id FROM production.stocks AS stc
                            WHERE stc.quantity > 1000 ))
```

Execution Plan Diagram



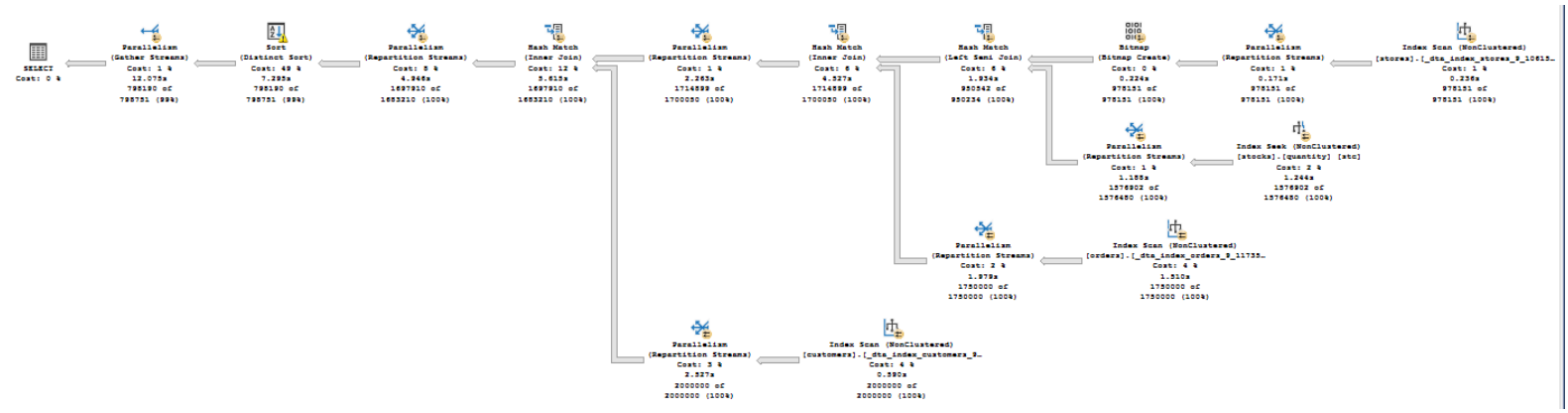
Some Analysis

Rows	StmtText	NodeId	Parent	PhysicalOp	LogicalOp	Argument	TotalSubtreeCost	Parallel
798190	SELECTDISTINCT c.first_name,c.last_nameFROMSales.orders AS o ,sales.customers AS c'	1	0				238.8167	False
798190	|--Sort(DISTINCT ORDER BY:([c].[first_name] ASC, [c].[last_name] ASC))	2	1	Sort	Distinct Sort	DISTINCT ORDER BY:([c].[first_name] ASC, [c].[last_name] ASC)	238.8167	False
1697910	|--Hash Match(Inner Join, HASH:([o].[customer_id])=([c].[customer_id]), RESIDUAL:(	4	2	Hash Match	Inner Join	HASH:([o].[customer_id])=([c].[customer_id]), RESIDUAL:([stor	130.845	False
1714899	|--Hash Match(Inner Join, HASH:([stc].[store_id])=([o].[store_id]))	6	4	Hash Match	Inner Join	HASH:([stc].[store_id])=([o].[store_id])	55.42873	False
950542	| |--Merge Join(Inner Join, MERGE:([sto].[store_id])=([stc].[store_id]), RESIDUAL	7	6	Merge Join	Inner Join	MERGE:([sto].[store_id])=([stc].[store_id]), RESIDUAL:([store].[	33.39322	False
978151	| | |--Clustered Index Scan(OBJECT:([store].[sales].[stores].[PK__stores__A2F2	8	7	Clustered Index Scan	Clustered Index Scan	OBJECT:([store].[sales].[stores].[PK__stores__A2F2A30CB0EFA4	20.97406	False
950542	| | |--Stream Aggregate(GROUP BY:([stc].[store_id]))	9	7	Stream Aggregate	Aggregate	GROUP BY:([stc].[store_id])	9.102805	False
1576902	| | |--Clustered Index Scan(OBJECT:([store].[production].[stocks].[PK__stocl	10	9	Clustered Index Scan	Clustered Index Scan	OBJECT:([store].[production].[stocks].[PK__stocks__E68284D39	7.197171	False
1750000	| |--Clustered Index Scan(OBJECT:([store].[sales].[orders].[PK__orders__465962	11	6	Clustered Index Scan	Clustered Index Scan	OBJECT:([store].[sales].[orders].[PK__orders__4659622960E1F0	7.539393	False
2000000	|--Clustered Index Scan(OBJECT:([store].[sales].[customers].[PK__customer__CD	12	4	Clustered Index Scan	Clustered Index Scan	OBJECT:([store].[sales].[customers].[PK__customer__CD65CB8	44.59291	False

SQL Query #1 Optimization

```
SELECT
		DISTINCT c.first_name,c.last_name
FROM
	sales.orders AS o INNER JOIN sales.customers AS c on   o.customer_id =
c.customer_id
WHERE
o.store_id IN (
	SELECT
		sto.store_id
	FROM
		sales.stores AS sto
WHERE
	sto.store_id IN ( SELECT stc.store_id FROM production.stocks AS stc
			WHERE stc.quantity > 1000 ))
```

Exection Plan Diagram



Some Analysis

Rows	StmtText	NodeId	Parent	PhysicalOp	LogicalOp	Argument	TotalSubtreeCost	Parallel
798190	SELECT DISTINCT c.first_name,c.last_name FROM sales.orders AS o INNER JOIN sales.cu	1	0				164.1221	False
798190	|--Parallelism(Gather Streams)	2	1	Parallelism	Gather Streams		164.1221	True
798190	|--Sort(DISTINCT ORDER BY:([c].[first_name] ASC, [c].[last_name] ASC))	3	2	Sort	Distinct Sort	DISTINCT ORDER BY:([c].[first_name] ASC, [c].[last_name] ASC)	162.7942	True
1697910	|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([c	4	3	Parallelism	Repartition Streams	PARTITION COLUMNS:([c].[first_name], [c].[last_name])	82.89161	True
1697910	|--Hash Match(Inner Join, HASH:([o].[customer_id])=([c].[customer_id]), RES	5	4	Hash Match	Inner Join	HASH:([o].[customer_id])=([c].[customer_id]), RESIDUAL:([new_store].[sales].[custom	69.05932	True
1714899	|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMN	6	5	Parallelism	Repartition Streams	PARTITION COLUMNS:([o].[customer_id])	38.04844	True
1714899	|--Hash Match(Inner Join, HASH:([sto].[store_id])=([o].[store_id]))	7	6	Hash Match	Inner Join	HASH:([sto].[store_id])=([o].[store_id])	35.98306	True
950542	|--Hash Match(Left Semi Join, HASH:([sto].[store_id])=([stc].[store_i	8	7	Hash Match	Left Semi Join	HASH:([sto].[store_id])=([stc].[store_id])	17.1488	True
978151	|--Bitmap(HASH:([sto].[store_id]), DEFINE:([Bitmap1004]))	9	8	Bitmap	Bitmap Create	HASH:([sto].[store_id])	2.63793	True
978151	|--Parallelism(Repartition Streams, Hash Partitioning, PARTITI	10	9	Parallelism	Repartition Streams	PARTITION COLUMNS:([sto].[store_id])	2.63793	True
978151	|--Index Scan(OBJECT:([new_store].[sales].[stores].[_dta_in	11	10	Index Scan	Index Scan	OBJECT:([new_store].[sales].[stores].[_dta_index_stores_9_1061578820__K1] AS [sto])	1.437483	True
1576902	|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION	12	8	Parallelism	Repartition Streams	PARTITION COLUMNS:([stc].[store_id])	5.390551	True
1576902	|--Index Seek(OBJECT:([new_store].[production].[stocks].[qua	13	12	Index Seek	Index Seek	OBJECT:([new_store].[production].[stocks].[quantity] AS [stc]), SEEK:([stc].[quantity] >	3.47323	True
1750000	|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION CO	14	7	Parallelism	Repartition Streams	PARTITION COLUMNS:([o].[store_id])	8.878751	True
1750000	|--Index Scan(OBJECT:([new_store].[sales].[orders].[_dta_index_c	15	14	Index Scan	Index Scan	OBJECT:([new_store].[sales].[orders].[_dta_index_orders_9_1173579219__K2_K7_1_3_	6.089407	True
2000000	|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMN	16	5	Parallelism	Repartition Streams	PARTITION COLUMNS:([c].[customer_id])	10.77584	True
2000000	|--Index Scan(OBJECT:([new_store].[sales].[customers].[_dta_index_cu	17	16	Index Scan	Index Scan	OBJECT:([new_store].[sales].[customers].[_dta_index_customers_9_1029578706__K1_	6.212092	True

What did we make to optimize?

[1] Query Optimization

- Make inner join on customer_id

[2] Adding Indexes

Index In	Columns	Sort Order
orders	Customer_id	ASC
	Store_id	ASC
stocks	quantity	ASC

[3] Parallel Execution

- The system has decided that it could benefit from performing some of the processing in parallel.
- It shouldn't be a cause for concern - generally - and the "duplicate" rows will be eliminated by the later (Gather Streams) operation.

NoSql Query #1

```
db.getCollection("dbo.ordcuststorstoc").aggregate([{
    $match: {
        quantity: {
            $gt: 1000
        },
        first_name: {
            $exists: true
        },
        last_name: {
            $exists: true
        }
    }
}, {
    $group: {
        _id: {
            first_name: "$first_name",
            last_name: "$last_name"
        }
    }
}, {
    $project: {
        first_name: "$_id.first_name",
        last_name: "$id.last_name"
    }
}, {
    $sort: {
        _id: - 1
    }
}], {
    "allowDiskUse": true
})
```

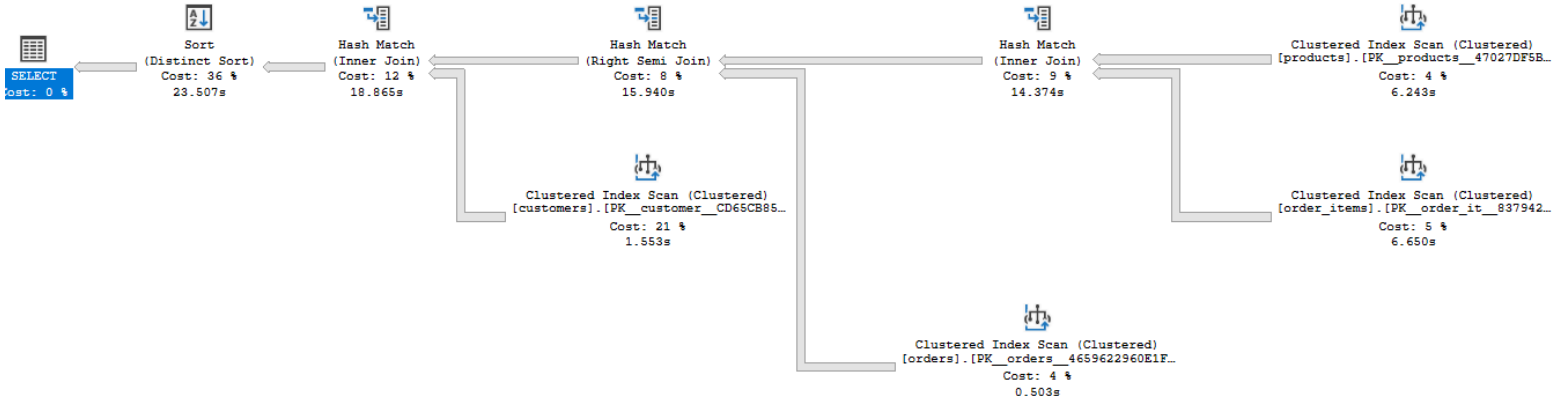
Time Analysis Query #1

Type	Time
SQL (Not Optimized)	00:48.472
Sql (Optimized)	00:22.594
NoSQL	00:67.676

SQL Query #2

```
SELECT
	DISTINCT c.first_name,
	c.last_name
FROM
	sales.orders AS o,
	sales.customers AS c
WHERE
	c.customer_id = o.customer_id
	AND o.shipped_date > '1970-08-11'
	AND o.order_id IN (
	SELECT
		oi.order_id
	FROM
		sales.order_items AS oi
	WHERE
	oi.discount > 10
	AND oi.product_id IN ( SELECT p.product_id FROM production.products AS p WHERE
	p.model_year > 1950 ));
```

Execution Plan Diagram



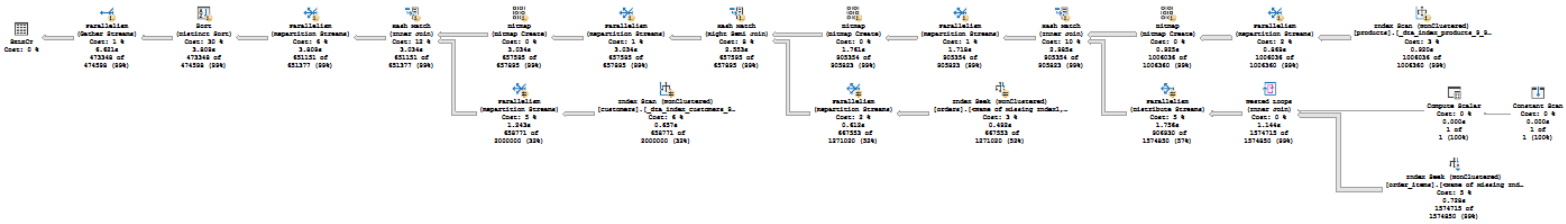
Some Analysis

Rows	StmtText	NodeId	Parent	PhysicalOp	LogicalOp	Argument	TotalSubtreeCost	Parallel
473348	SELECT DISTINCT c.first_name, c.last_name FROM sales.orders AS o, sales.customers AS	1	0				211.2786	False
473348	|--Sort(DISTINCT ORDER BY:([c].[first_name] ASC, [c].[last_name] ASC))	2	1	Sort	Distinct Sort	DISTINCT ORDER BY:([c].[first_name] ASC, [c].[last_name] ASC)	211.2786	False
651151	|--Hash Match(Inner Join, HASH:([o].[customer_id])=([c].[customer_id]), RESIDUA	3	2	Hash Match	Inner Join	HASH:([o].[customer_id])=([c].[customer_id]), RESIDUAL:([store].[sales].[customers].	135.9496	False
657595	|--Hash Match(Right Semi Join, HASH:([oi].[order_id])=([o].[order_id]))	4	3	Hash Match	Right Semi Join	HASH:([oi].[order_id])=([o].[order_id])	65.59115	False
905354	| |--Hash Match(Inner Join, HASH:([p].[product_id])=([oi].[product_id]))	6	4	Hash Match	Inner Join	HASH:([p].[product_id])=([oi].[product_id])	40.18327	False
1006036	| | |--Clustered Index Scan(OBJECT:([store].[production].[products].[PK__p	7	6	Clustered Index Scan	Clustered Index Scan	OBJECT:([store].[production].[products].[PK__products__47027DF5B1341017] AS [p]),	9.39643	False
1574715	| | |--Clustered Index Scan(OBJECT:([store].[sales].[order_items].[PK__orde	8	6	Clustered Index Scan	Clustered Index Scan	OBJECT:([store].[sales].[order_items].[PK__order_it__837942D42157409F] AS [oi]), WI	11.24458	False
1270917	| |--Clustered Index Scan(OBJECT:([store].[sales].[orders].[PK__orders__4659	9	4	Clustered Index Scan	Clustered Index Scan	OBJECT:([store].[sales].[orders].[PK__orders__4659622960E1F0CB] AS [o]), WHERE:([st	7.539393	False
2000000	|--Clustered Index Scan(OBJECT:([store].[sales].[customers].[PK__customer__	10	3	Clustered Index Scan	Clustered Index Scan	OBJECT:([store].[sales].[customers].[PK__customer__CD65CB8567E6237E] AS [c])	44.59291	False

SQL Query #2 Optimization

```
SELECT
	DISTINCT c.first_name,
	c.last_name
FROM
	( SELECT so.order_id, so.customer_id FROM sales.orders AS so WHERE
so.shipped_date > '1970-08-11' ) AS o
	INNER JOIN sales.customers AS c ON c.customer_id = o.customer_id
WHERE
	o.order_id IN (
	SELECT
		oi.order_id
	FROM
		sales.order_items AS oi
	WHERE
	oi.discount > 10
	AND oi.product_id IN ( SELECT p.product_id FROM production.products AS p WHERE
	p.model_year > 1950 ))
```

Exection Plan Diagram



Some Analysis

Rows	StmtText	NodeId	Parent	PhysicalOp	LogicalOp	Argument	TotalSubtreeCost	Parallel
473348	SELECT DISTINCT c.first_name, c.last_name FROM SELECT	1	0				95.97579	False
473348	|--Parallelism(Gather Streams)	2	1	Parallelism	Gather Streams		95.97579	True
473348	|--Sort(DISTINCT ORDER BY:([c].[first_name] ASC, [c].	3	2	Sort	Distinct Sort	DISTINCT ORDER BY:([c].[first_name] ASC, [c].[last_name] ASC)	95.17524	True
651151	|--Parallelism(Repartition Streams, Hash Partitioni	4	3	Parallelism	Repartition Streams	PARTITION COLUMNS:([c].[first_name], [c].[last_name])	66.29818	True
651151	|--Hash Match(Inner Join, HASH:([so].[customer	5	4	Hash Match	Inner Join	HASH:([so].[customer_id])=([c].[customer_id]), RESIDUAL:([new_	60.92782	True
657595	|--Bitmap(HASH:([so].[customer_id]), DEFINE	6	5	Bitmap	Bitmap Create	HASH:([so].[customer_id])	38.30396	True
657595	|--Parallelism(Repartition Streams, Hash P	7	6	Parallelism	Repartition Streams	PARTITION COLUMNS:([so].[customer_id])	38.30396	True
657595	|--Hash Match(Right Semi Join, HASH:([	8	7	Hash Match	Right Semi Join	HASH:([oi].[order_id])=([so].[order_id])	37.48721	True
905354	|--Bitmap(HASH:([oi].[order_id]), DEF	9	8	Bitmap	Bitmap Create	HASH:([oi].[order_id])	24.92697	True
905354	|--Parallelism(Repartition Streams	11	9	Parallelism	Repartition Streams	PARTITION COLUMNS:([oi].[order_id])	24.92697	True
905354	|--Hash Match(Inner Join, HASH	12	11	Hash Match	Inner Join	HASH:([p].[product_id])=([oi].[product_id])	23.81318	True
1006036	|--Bitmap(HASH:([p].[product	13	12	Bitmap	Bitmap Create	HASH:([p].[product_id])	4.54513	True
1006036	|--Parallelism(Repartition	14	13	Parallelism	Repartition Streams	PARTITION COLUMNS:([p].[product_id])	4.54513	True
1006036	|--Index Scan(OBJECT:([	15	14	Index Scan	Index Scan	OBJECT:([new_store].[production].[products].[_dta_index_produ	2.890889	True
906930	|--Parallelism(Distribute Stre	16	12	Parallelism	Distribute Streams	PARTITION COLUMNS:([oi].[product_id]), WHERE:(PROBE([Bitma	10.04797	True
1574715	|--Nested Loops(Inner Join	17	16	Nested Loops	Inner Join	OUTER REFERENCES:([Expr1005], [Expr1006], [Expr1004])	5.050431	False
1	|--Compute Scalar(DEFIN	18	17	Compute Scalar	Compute Scalar	DEFINE:(([Expr1005],[Expr1006],[Expr1004])=GetRangeWithMism	0	False
1	|--Constant Scan	19	18	Constant Scan	Constant Scan		0	False
1574715	|--Index Seek(OBJECT:([	20	17	Index Seek	Index Seek	OBJECT:([new_store].[sales].[order_items].[<Name of Missing In	5.050431	False
667553	|--Parallelism(Repartition Streams, H	21	8	Parallelism	Repartition Streams	PARTITION COLUMNS:([so].[order_id])	4.715209	True
667553	|--Index Seek(OBJECT:([new_store	22	21	Index Seek	Index Seek	OBJECT:([new_store].[sales].[orders].[<Name of Missing Index1,	2.681522	True
658771	|--Parallelism(Repartition Streams, Hash Part	23	5	Parallelism	Repartition Streams	PARTITION COLUMNS:([c].[customer_id])	10.77584	True
658771	|--Index Scan(OBJECT:([new_store].[sales]	24	23	Index Scan	Index Scan	OBJECT:([new_store].[sales].[customers].[_dta_index_customer	6.212092	True

What did we make to optimize?

[1] Query Optimization

- Select first on the order.shipped_date and return only the columns needed later
- Make inner join on customer_id of order and customer table

[2] Adding Indexes

Index In	Columns	Sort Order
products	protduct_id	ASC
	model_year	ASC
order_items	discount	ASC
orders	shipped_date	ASC

[3] Parallel Execution

- The system has decided that it could benefit from performing some of the processing in parallel.
- It shouldn't be a cause for concern - generally - and the "duplicate" rows will be eliminated by the later (Gather Streams) operation.

NoSql Query #2

```
db.getCollection("dbo.ordcusitempro").aggregate([{
    $match: {
        model_year: {
            $gt: 1950
        },
                        shipped_date: {
            $gt: new ISODate('1970-08-11')
        },
                        discount: {
            $gt: 10
        },
        first_name: {
            $exists: true
        },
        last_name: {
            $exists: true
        }
    }
}, {
    $group: {
        _id: {
            first_name: "$first_name",
            last_name: "$last_name"
        }
    }
}, {
```

```
    $project: {
        first_name: "$_id.first_name",
        last_name: "$id.last_name"
    }
}, {
    $sort: {
        _id:  - 1
    }
}], {
    "allowDiskUse": true
})
```

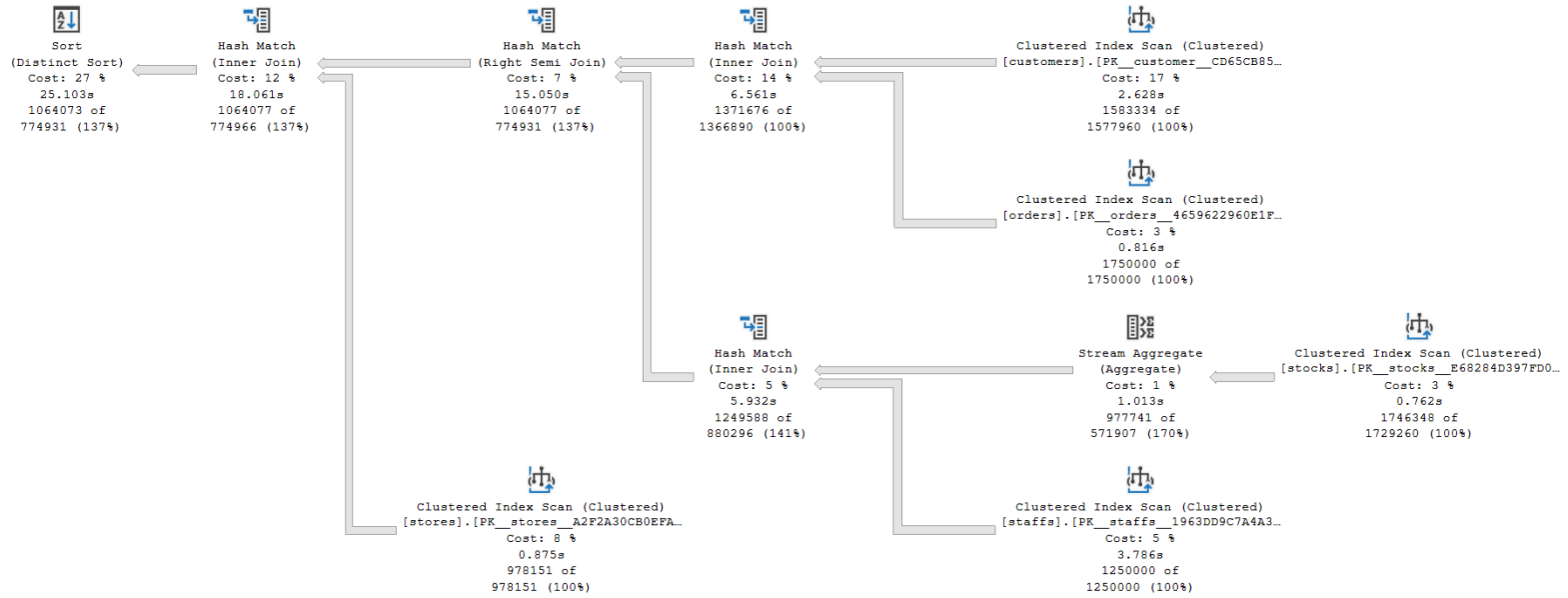
Time Analysis Query #2

Type	Time
SQL (Not Optimized)	00:32.914
Sql (Optimized)	00:14.589
NoSQL	00:58.487

SQL Query #3

```
SELECT
	DISTINCT s.first_name,
	s.last_name,
	sto.store_name
FROM
	sales.staffs AS s,
	sales.stores AS sto
WHERE
	s.store_id = sto.store_id
	AND sto.store_id IN ( SELECT st.store_id FROM production.stocks AS st
WHERE st.quantity > 20 )
	AND s.staff_id IN (
	SELECT
		o.staff_id
	FROM
		sales.orders AS o
	WHERE
	o.staff_id = s.staff_id
	AND o.customer_id IN ( SELECT c.customer_id FROM sales.customers AS c WHERE
	c.zip_code > '2000' ))
```

Execution Plan Diagram



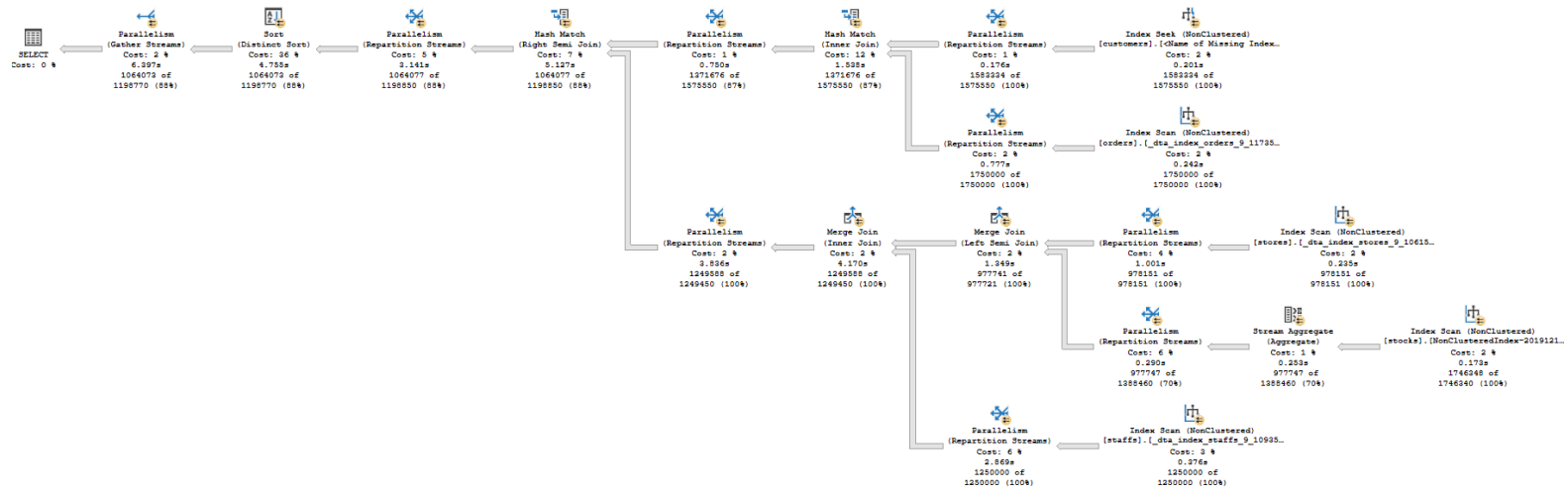
Some Analysis

Rows	StmtText	NodeId	Parent	PhysicalOp	LogicalOp	Argument	TotalSubtreeCost	Parallel
1064073	SELECT DISTINCT s.first_name, s.last_name, sto.store_name FROM	1	0				261.471	False
1064073	|--Sort(DISTINCT ORDER BY:([s].[first_name] ASC, [s].[last_nam	2	1	Sort	Distinct Sort	DISTINCT ORDER BY:([s].[first_name] ASC, [s].[last_name] A	261.471	False
1064077	|--Hash Match(Inner Join, HASH:([st].[store_id])=([sto].[stor	4	2	Hash Match	Inner Join	HASH:([st].[store_id])=([sto].[store_id])	191.8728	False
1064077	|--Hash Match(Right Semi Join, HASH:([o].[staff_id])=([s].[	5	4	Hash Match	Right Semi Join	HASH:([o].[staff_id])=([s].[staff_id]), RESIDUAL:([store].[sale	139.2443	False
1371676	| |--Hash Match(Inner Join, HASH:([c].[customer_id])=([c	7	5	Hash Match	Inner Join	HASH:([c].[customer_id])=([o].[customer_id]), RESIDUAL:([s	88.02207	False
1583334	| | |--Clustered Index Scan(OBJECT:([store].[sales].[cus	8	7	Clustered Index Scan	Clustered Index Scan	OBJECT:([store].[sales].[customers].[PK__customer__CD65C	44.59291	False
1750000	| | |--Clustered Index Scan(OBJECT:([store].[sales].[orc	9	7	Clustered Index Scan	Clustered Index Scan	OBJECT:([store].[sales].[orders].[PK__orders__4659622960E	7.539393	False
1249588	| |--Hash Match(Inner Join, HASH:([st].[store_id])=([s].[s	10	5	Hash Match	Inner Join	HASH:([st].[store_id])=([s].[store_id])	33.21813	False
977741	| |--Stream Aggregate(GROUP BY:([st].[store_id]))	11	10	Stream Aggregate	Aggregate	GROUP BY:([st].[store_id])	9.187757	False
1746348	| | |--Clustered Index Scan(OBJECT:([store].[product	12	11	Clustered Index Scan	Clustered Index Scan	OBJECT:([store].[production].[stocks].[PK__stocks__E68284[	7.197171	False
1250000	| |--Clustered Index Scan(OBJECT:([store].[sales].[staf	13	10	Clustered Index Scan	Clustered Index Scan	OBJECT:([store].[sales].[staffs].[PK__staffs__1963DD9C7A4A	12.04347	False
978151	|--Clustered Index Scan(OBJECT:([store].[sales].[stores].[F	14	4	Clustered Index Scan	Clustered Index Scan	OBJECT:([store].[sales].[stores].[PK__stores__A2F2A30CB0E	20.97406	False

SQL Query #3 Optimization

```
SELECT DISTINCT
	s.first_name,
	s.last_name,
	sto.store_name
FROM
	sales.staffs AS s
	INNER JOIN sales.stores AS sto ON sto.store_id = s.store_id
WHERE
	sto.store_id IN ( SELECT st.store_id FROM production.stocks AS st WHERE
st.quantity > 20 )
	AND s.staff_id IN (
	SELECT
		o.staff_id
	FROM
		sales.orders AS o
	WHERE o.customer_id IN ( SELECT c.customer_id FROM sales.customers AS c
					WHERE c.zip_code > '2000' ))
```

Exection Plan Diagram



Some Analysis

Rows	StmtText	NodeId	Parent	PhysicalOp	LogicalOp	Argument	TotalSubtreeCost	Parallel
1064073	SELECT DISTINCT [first_name],[last_name],[sto.store_name FROM[Sale	1	0				156.2407	False
1064073	|--Parallelism(Gather Streams)	2	1	Parallelism	Gather Streams		156.2407	True
1064073	|--Sort(DISTINCT ORDER BY:([s].[first_name] ASC, [s].[last_name]	3	2	Sort	Distinct Sort	DISTINCT ORDER BY:([s].[first_name] ASC, [s].[last_name] ASC, [sto].[	153.5176	True
1064077	|--Parallelism(Repartition Streams, Hash Partitioning, PARTITIO	4	3	Parallelism	Repartition Streams	PARTITION COLUMNS:([s].[first_name], [s].[last_name], [sto].[store_	97.95304	True
1064077	|--Hash Match(Right Semi Join, HASH:([o].[staff_id])=([s].[sta	5	4	Hash Match	Right Semi Join	HASH:([o].[staff_id])=([s].[staff_id])	89.58683	True
1371676	|--Parallelism(Repartition Streams, Hash Partitioning, PAR	7	5	Parallelism	Repartition Streams	PARTITION COLUMNS:([o].[staff_id])	32.50592	True
1371676	|--Hash Match(Inner Join, HASH:([c].[customer_id])=([c	8	7	Hash Match	Inner Join	HASH:([c].[customer_id])=([o].[customer_id]), RESIDUAL:([new_store	30.58972	True
1583334	|--Parallelism(Repartition Streams, Hash Partitioning	9	8	Parallelism	Repartition Streams	PARTITION COLUMNS:([c].[customer_id])	5.513364	True
1583334	|--Index Seek(OBJECT:([new_store].[sales].[custo	10	9	Index Seek	Index Seek	OBJECT:([new_store].[sales].[customers].[<Name of Missing Index, sy	3.597162	True
1750000	|--Parallelism(Repartition Streams, Hash Partitioning	11	8	Parallelism	Repartition Streams	PARTITION COLUMNS:([o].[customer_id])	6.643196	True
1750000	|--Index Scan(OBJECT:([new_store].[sales].[order	12	11	Index Scan	Index Scan	OBJECT:([new_store].[sales].[orders].[_dta_index_orders_9_1173579	3.853852	True
1249588	|--Parallelism(Repartition Streams, Hash Partitioning, PAR	13	5	Parallelism	Repartition Streams	PARTITION COLUMNS:([s].[staff_id])	45.60593	True
1249588	|--Merge Join(Inner Join, MERGE:([sto].[store_id])=([s].	14	13	Merge Join	Inner Join	MERGE:([sto].[store_id])=([s].[store_id]), RESIDUAL:([new_store].[sal	41.96824	True
977741	|--Merge Join(Left Semi Join, MERGE:([sto].[store_id	15	14	Merge Join	Left Semi Join	MERGE:([sto].[store_id])=([st].[store_id]), RESIDUAL:([new_store].[sa	25.37505	True
978151	|--Parallelism(Repartition Streams, Hash Partition	16	15	Parallelism	Repartition Streams	PARTITION COLUMNS:([sto].[store_id]), ORDER BY:([sto].[store_id] AS	9.147228	True
978151	|--Index Scan(OBJECT:([new_store].[sales].[sto	17	16	Index Scan	Index Scan	OBJECT:([new_store].[sales].[stores].[_dta_index_stores_9_10615788	2.448594	True
977747	|--Parallelism(Repartition Streams, Hash Partition	18	15	Parallelism	Repartition Streams	PARTITION COLUMNS:([st].[store_id]), ORDER BY:([st].[store_id] ASC)	13.69118	True
977747	|--Stream Aggregate(GROUP BY:([st].[store_id])	19	18	Stream Aggregate	Aggregate	GROUP BY:([st].[store_id])	5.056811	True
1746348	|--Index Scan(OBJECT:([new_store].[producti	20	19	Index Scan	Index Scan	OBJECT:([new_store].[production].[stocks].[NonClusteredIndex-2019	3.853111	True
1250000	|--Parallelism(Repartition Streams, Hash Partitioning	21	14	Parallelism	Repartition Streams	PARTITION COLUMNS:([s].[store_id]), ORDER BY:([s].[store_id] ASC)	14.1888	True
1250000	|--Index Scan(OBJECT:([new_store].[sales].[staffs]	22	21	Index Scan	Index Scan	OBJECT:([new_store].[sales].[staffs].[_dta_index_staffs_9_109357893	4.515148	True

What did we make to optimize?

[1] Query Optimization

- Make inner join on store_id of staff and store table

[2] Adding Indexes

Index In	Columns	Sort Order
staff	staff_id	ASC
	store_id	ASC
stocks	quantity	ASC
	store_id	ASC
orders	customer_id	ASC
	statff_id	ASC
customers	zip_code	ASC

[3] Parallel Execution

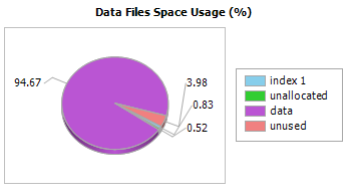
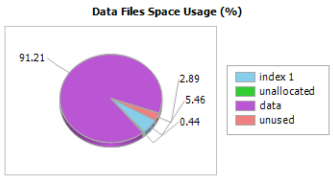
- The system has decided that it could benefit from performing some of the processing in parallel.
- It shouldn't be a cause for concern - generally - and the "duplicate" rows will be eliminated by the later (Gather Streams) operation.

NoSql Query #3

```
db.getCollection("dbo.stafstostocordcus").aggregate([{
    $match: {
        quantity: {
            $gt: 20
        },
        zip_code: {
            $gt: "2000"
        },
        store_name: {
            $exists: true
        },
        s_first_name: {
            $exists: true
        },
        s_last_name: {
            $exists: true
        }
    }
}, {
    $group: {
        _id: {
            store_name: "$store_name",
            s_first_name: "$s_first_name",
            s_last_name: "$s_last_name"
        }
    }
}, {
    $project: {
        store_name: "$_id.store_name",
        s_first_name: "$_id.s_first_name",
        s_last_name: "$id.s_last_name"
    }
}, {
    $sort: {
        _id: - 1
    }
}], {
    "allowDiskUse": true
})
```

Time Analysis Query #3

Type	Time
SQL (Not Optimized)	01:05.357
Sql (Optimized)	00:22.786
NoSQL	00:89.435

Name	Old / NON-Optimized DB	New / Optimized DB
Data File Space Reserved	11,542,25 MB	16,086,25 MB
Data File Space Usage		
Query #1 Rows	798,190	
Query #2 Rows	473,348	
Query #3 Rows	1,064,073	
Query #1 Time	00:48.472	00:22.594
Query #2 Time	00:32.914	00:14.589
Query #3 Time	01:05.357	00:22.786
Time Optimized Query#1	+53%	
Time Optimized Query#2	+56%	
Time Optimized Query#3	+65%	
Avg. Time Optimization %	+58%	
Space Degradation	-39%	
On Another HW	RAM 16 GB: is a lot better because DB doesn't need to put the records that came on disk when the ram gets near to full.	
Comments (Recommendation and Opinion)	1. use HW faster in CPU performance than this so he can make the comparisons more easily. 2. use HW has larger ram than this so DB doesn't need to put the records that came on disk when the ram gets near to full. 3. use stored procedures for faster execution. 4. change some in the schema, ex: you don't need to put the customer image column on a separate table you can combine it with the customers table. 5. put images on the disk and take a reference for it in the customer image column. 6. remove some un-needed index to optimize the space.	