

**PES UNIVERSITY**  
**ELECTIVE 1: DATABASE TECHNOLOGIES (UE18CS315)**  
**ASSIGNMENT 1**

**NAME:** SHAAZIN SHEIKH SHUKOOR

**SRN:** PES1201801754

**SEMESTER:** 5

**SECTION:** J

**I. Review the data model of the project you had submitted in lieu of ISA-2 in 4<sup>th</sup> semester. Identify gaps in the steps followed and revise the model accordingly.**

**INTRODUCTION**

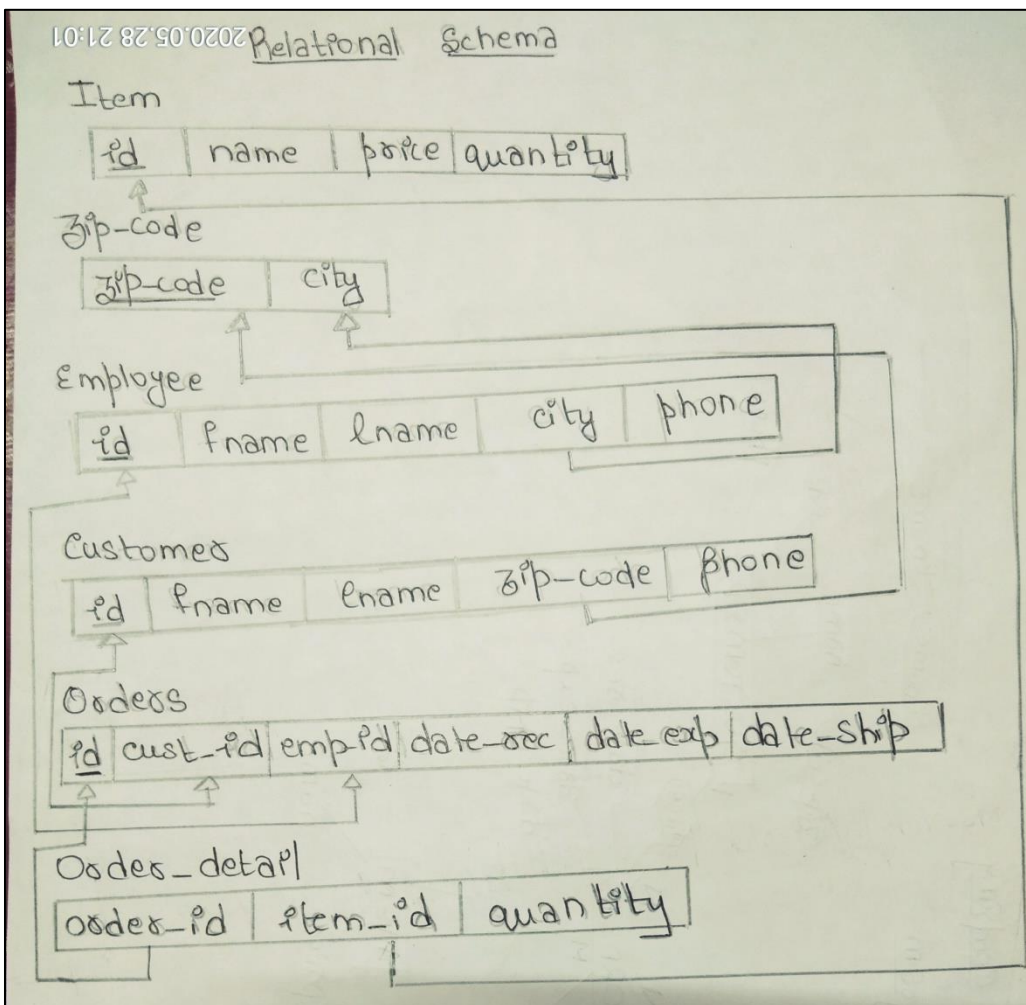
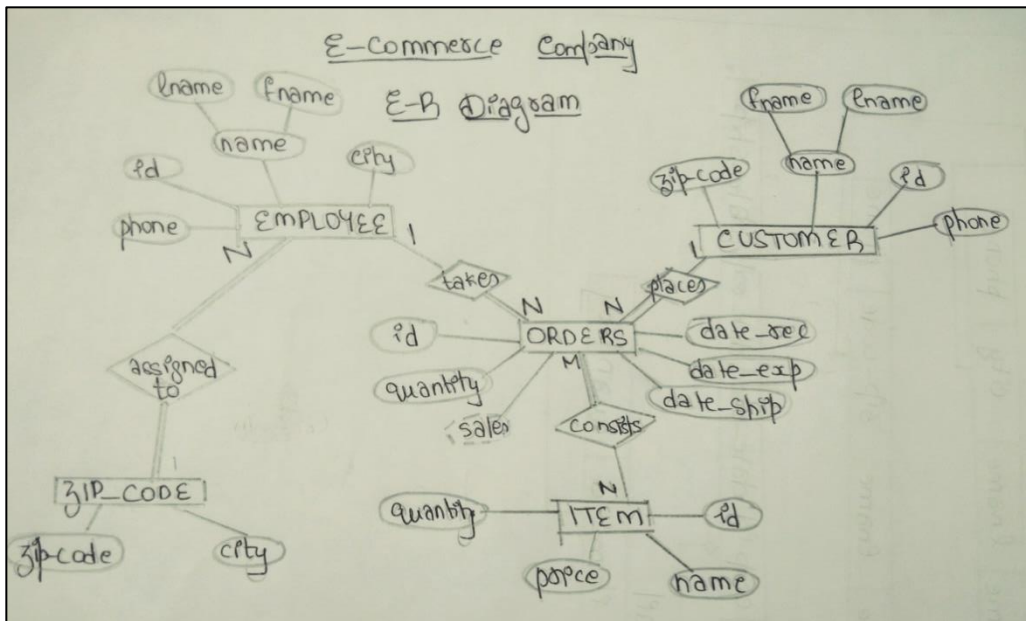
**DATABASE: E-COMMERCE COMPANY**

The E-commerce company needs to store information about Item(identified by id, name, price, quantity as attributes), Zip\_code(identified by zip\_code, city as attributes), Employee(identified by id, fname, lname, city, phone as attributes), Customer(identified by id, fname, lname, zip\_code, phone as attributes), Orders(identified by id, cust\_id, emp\_id, date\_rec, date\_exp, date\_ship as attributes) and Order\_detail(identified by order\_id, item\_id, quantity as attributes).

Orders belong to a category, employee *takes* order, customer *places* an order, order *consists of* items, employee *assigned to* zip\_code.

An employee can take many orders or may not take any orders at all. An order is assigned to only one employee. A customer can place many or no orders and an order is placed by one customer. One order can consist of many items and one item can be present in many orders. All employees are assigned to a zip\_code(city) and all zip\_codes have one or more employees assigned to it.

## ER DIAGRAM AND RELATIONAL SCHEMA



## **FUNCTIONAL DEPENDENCIES AND CHOICE OF KEYS**

Closure properties were applied to all the attributes to find the candidate keys of all relations. In case of multiple candidate keys, one was chosen as the primary key.

For example in item relation:

$\{id\}^+ = \{id, name, price, quantity\}$

As closure of id is the set of all attributes of the relation, id hence can be considered as a candidate key.

### **ITEM relation:**

Id->name, price, quantity

Primary key: id

### **ZIP\_CODE relation:**

Zip\_code->city

Primary key: zip\_code

### **EMPLOYEE relation:**

Id->fname, lname, city, phone

Primary key: id

Foreign key: city(zip\_code)

### **CUSTOMER relation:**

Id-> fname, lname, zip\_code, phone

Phone->id, fname, lname, zip\_code (assuming phone nos are not repeated)

Primary key: id

Foreign key: zip\_code(zip\_code)

### **ORDERS relation:**

Id->cust\_id, emp\_id, date\_rec, date\_exp, date\_ship

Primary key: id

Foreign key: cust\_id(customer), emp\_id(employee)

### **ORDER\_DETAIL relation:**

Order\_id, item\_id -> quantity

Primary key: order\_id and item\_id

Foreign key: order\_id(orders), item-id(item)

## NORMALIZATION AND TESTING FOR LOSSLESS JOIN

All the relations in the database schema obtained from the ER diagram are in 3NF and BCNF as there exists no transitive dependencies in any of the relations. However, 3NF may be violated when upon the addition of an attribute; there exists a dependency from a non-prime to non-prime attribute.

For example, if the attribute 'city' is added to Customer relation, there is a new dependency from zip\_code -> city which leads to a transitive dependency thereby violating 3NF. Another example is when 'city' is added to Orders relation. There is a dependency from emp\_id -> city, state which violates 3NF.

A 2nd NF will be violated when there exists a partial dependency from a non-prime attribute to parts of a candidate key. 2nd NF will be violated if we add 'item\_name' to Order\_detail relation, because item\_id->item\_name but the key is item\_id, order\_id, which is a partial dependency thereby violating 2nd NF.

A decomposition DECOMP = {R1, R2, . . . , Rm} of R has the lossless join property with respect to the set of dependencies F on R if, for every relation state r of R that satisfies F, the following holds, where \* is the NATURAL JOIN of all the relations in DECOMP:

$$*(\pi R_1(r), \dots, \pi R_m(r)) = r$$

**Using Chase's algorithm:**

	id	name	price	quantity	zip_code	city	eid	efname	elname	ephone	cid	cfname	clname	cphone	oid	rec	exp	ship	quantity
item	b11	b12	b13	b14	b15	b16	b17	b18	b19	b110	b111	b112	b113	b114	b115	b116	b117	b118	b119
zip_code	b21	b22	b23	b24	b25	b26	b27	b28	b29	b210	b211	b212	b213	b214	b215	b216	b217	b218	b219
employee	b31	b32	b33	b34	b35	b36	b37	b38	b39	b310	b311	b312	b313	b314	b315	b316	b317	b318	b319
customer	b41	b42	b43	b44	b45	b46	b47	b48	b49	b410	b411	b412	b413	b414	b415	b416	b417	b418	b419
orders	b51	b52	b53	b54	b55	b56	b57	b58	b59	b510	b511	b512	b513	b514	b515	b516	b517	b518	b519
order_detail	b61	b62	b63	b64	b65	b66	b67	b68	b69	b610	b611	b612	b613	b614	b615	b616	b617	b618	b619
	id	name	price	quantity	zip_code	city	eid	efname	elname	ephone	cid	cfname	clname	cphone	oid	rec	exp	ship	quantity
item	a1	a2	a3	a4	b15	b16	b17	b18	b19	b110	b111	b112	b113	b114	b115	b116	b117	b118	b119
zip_code	b21	b22	b23	b24	a5	a6	b27	b28	b29	b210	b211	b212	b213	b214	b215	b216	b217	b218	b219
employee	b31	b32	b33	b34	b35	a6	a7	a8	a9	a10	b311	b312	b313	b314	b315	b316	b317	b318	b319
customer	b41	b42	b43	b44	a14	b46	b47	b48	b49	b410	a11	a12	a13	a14	b415	b416	b417	b418	b419
orders	b51	b52	b53	b54	b55	b56	a7	b58	b59	b510	a11	b512	b513	b514	a15	a16	a17	a18	b519
order_detail	a1	b62	b63	b64	b65	b66	b67	b68	b69	b610	b611	b612	b613	b614	a15	b616	b617	b618	a19
	id	name	price	quantity	zip_code	city	eid	efname	elname	ephone	cid	cfname	clname	cphone	oid	rec	exp	ship	quantity
item	a1	a2	a3	a4	b15	b16	b17	b18	b19	b110	b111	b112	b113	b114	b115	b116	b117	b118	b119
zip_code	b21	b22	b23	b24	a5	a6	b27	b28	b29	b210	b211	b212	b213	b214	b215	b216	b217	b218	b219
employee	b31	b32	b33	b34	b35	a6	a7	a8	a9	a10	b311	b312	b313	b314	b315	b316	b317	b318	b319
customer	b41	b42	b43	b44	a5	a6	b47	b48	b49	b410	a11	a12	a13	a14	b415	b416	b417	b418	b419
orders	b51	b52	b53	b54	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14	a15	a16	a17	a18	b519
order_detail	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14	a15	a16	a17	a18	a19

Since all the attributes of the order\_detail has 'a' values (i.e., the last row), it is verified that the above relation has lossless join property.

## II. Review memory management in any of the contemporary DBMS

Using the below commands, the memory managed by Microsoft SQL Server can be understood. On executing, we see how different pools for different types of data are created and managed.

```
sp_configure
--based on the server memory, it shows how pools are allocated to store data

select * from sys.dm_os_sys_memory
--shows the total memory and memory allocated by each of the pools

select * from sys.dm_os_memory_objects
--memory object and address in each of the pools
--block size is 8K (page size in bytes)

select * from sys.dm_os_process_memory
--shows os memory, server memory, virtual memory, counters related to memory management
--by os

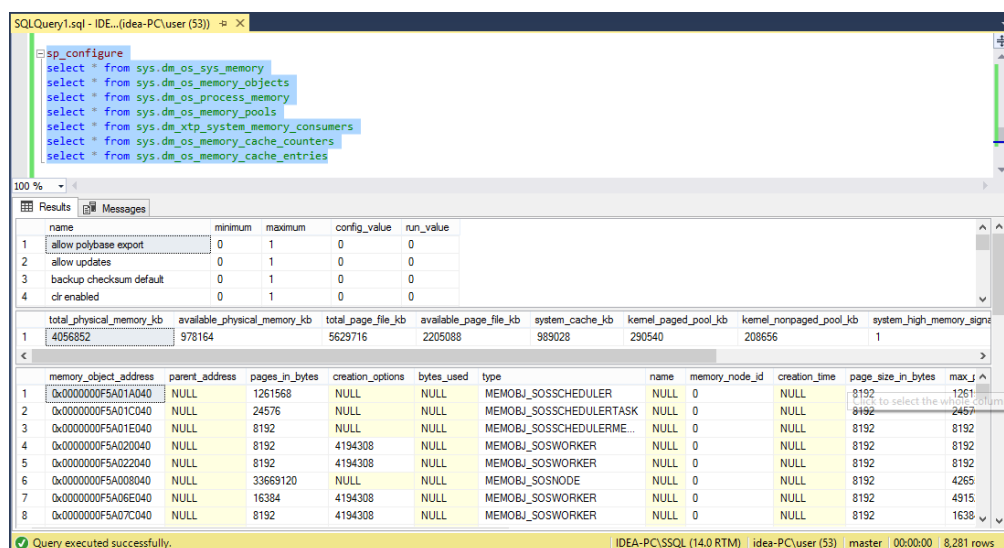
select * from sys.dm_os_memory_pools
--operating system memory pools, pools created to store os related objects

select * from sys.dm_xtp_system_memory_consumers
--showshow much and what is each consumer using, how much allocated and how much is being
--used

select * from sys.dm_os_memory_cache_counters
--how much page(=block) is allocated by each of the objects

select * from sys.dm_os_memory_cache_entries
--if something is present in cache, how much memory is being used
```

### Execution:



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows the execution of the `sp_configure` command, which has returned a table of configuration settings. The bottom pane shows the results of the `sys.dm_os_memory_pools` query, which lists various memory pools and their associated statistics.

name	minimum	maximum	config_value	run_value
allow polybase export	0	1	0	0
allow updates	0	1	0	0
backup checksum default	0	1	0	0
clr enabled	0	1	0	0

total_physical_memory_kb	available_physical_memory_kb	total_page_file_kb	available_page_file_kb	system_cache_kb	kemmel_paged_pool_kb	kemmel_nonpaged_pool_kb	system_high_memory_sign
4056852	978154	5629716	2205088	989028	290540	208656	1

memory_object_address	parent_address	pages_in_bytes	creation_options	bytes_used	type	name	memory_node_id	creation_time	page_size_in_bytes	max...
0x0000000F5A01A040	NULL	1261568	NULL	NULL	MEMOBJ_SOSSCHEDULER	NULL	0	NULL	8192	1261
0x0000000F5A01C040	NULL	24576	NULL	NULL	MEMOBJ_SOSSCHEDULERTASK	NULL	0	NULL	8192	2457
0x0000000F5A01E040	NULL	8192	NULL	NULL	MEMOBJ_SOSSCHEDULERME...	NULL	0	NULL	8192	8192
0x0000000F5A020040	NULL	8192	4194308	NULL	MEMOBJ_SOSWORKER	NULL	0	NULL	8192	8192
0x0000000F5A022040	NULL	8192	4194308	NULL	MEMOBJ_SOSWORKER	NULL	0	NULL	8192	8192
0x0000000F5A008040	NULL	33669120	NULL	NULL	MEMOBJ_SOSNODE	NULL	0	NULL	8192	4265
0x0000000F5A06E040	NULL	16384	4194308	NULL	MEMOBJ_SOSWORKER	NULL	0	NULL	8192	4915
0x0000000F5A07C040	NULL	8192	4194308	NULL	MEMOBJ_SOSWORKER	NULL	0	NULL	8192	1638

SQLQuery1.sql - IDE...(idea-PC\user (53))

```

sp_configure
select * from sys.dm_os_sys_memory
select * from sys.dm_os_memory_objects
select * from sys.dm_os_process_memory
select * from sys.dm_os_memory_pools
select * from sys.dm_xtp_system_memory_consumers
select * from sys.dm_os_memory_cache_counters
select * from sys.dm_os_memory_cache_entries

```

100 %

Results Messages

	physical_memory_in_use_kb	large_page_allocations_kb	locked_page_allocations_kb	total_virtual_address_space_kb	virtual_address_space_reserved_kb	virtual_address_space_committed_kb	virtual_
1	358416	0	0	137438953344	10880736	523904	13742

	memory_pool_address	pool_id	type	name	max_free_entries_count	free_entries_count	removed_in_all_rounds_count
1	0x0000000F5A040040	0	OBJECTSTORE_SNI_PACKET	SNI Packet	1024	0	0
2	0x0000000F5A040E90	0	OBJECTSTORE_SNI_PACKET	SNI Packet	1024	4	0
3	0x0000000F5A046040	0	OBJECTSTORE_SNI_PACKET	SNI Packet	1024	2	0
4	0x0000000F5A046E90	0	OBJECTSTORE_SNI_PACKET	SNI Packet	1024	0	0
5	0x0000000F5A04C040	0	OBJECTSTORE_SNI_PACKET	SNI Packet	1024	0	0
6	0x0000000F5A04CE90	0	OBJECTSTORE_SNI_PACKET	SNI Packet	1024	0	0
7	0x0000000F5A052040	0	OBJECTSTORE_SNI_PACKET	SNI Packet	1024	0	0
8	0x0000000F5A052E90	0	OBJECTSTORE_SNI_PACKET	SNI Packet	1024	0	0

	memory_consumer_id	memory_consumer_type	memory_consumer_type_desc	memory_consumer_desc	lookaside_id	allocated_bytes	used_bytes	allocation_count	partition_count	sizeclass_co
1	29	2	VARHEAP	Lookaside heap	NULL	0	0	0	4	73
2	28	4	PGPOOL	256K page pool	NULL	0	0	0	1	1
3	27	4	PGPOOL	4K page pool	NULL	0	0	0	1	1
4	26	2	VARHEAP	System heap	NULL	458752	413888	2467	4	112

Query executed successfully. IDEA-PC\SSQL (14.0 RTM) idea-PC\user (53) master 00:00:00 8,281 rows

SQLQuery1.sql - IDE...(idea-PC\user (53))

```

sp_configure
select * from sys.dm_os_sys_memory
select * from sys.dm_os_memory_objects
select * from sys.dm_os_process_memory
select * from sys.dm_os_memory_pools
select * from sys.dm_xtp_system_memory_consumers
select * from sys.dm_os_memory_cache_counters
select * from sys.dm_os_memory_cache_entries

```

100 %

Results Messages

	cache_address	name	type	pages_kb	pages_in_use_kb	entries_count	entries_in_use_count
16	0x0000000F58A8E080	Service Broker user c...	CACHESTORE_BROK...	8	0	0	0
17	0x0000000F58A8ECD0	Service Broker Null R...	CACHESTORE_BROK...	8	0	0	0
18	0x0000000F58A82080	Service Broker Trans...	CACHESTORE_BROK...	8	0	0	0
19	0x0000000F58ABAED0	Broker dormant rowsets	CACHESTORE_BROK...	16	0	0	0
20	0x0000000F58AC6080	Broker dormant rowsets	CACHESTORE_BROK...	8	0	0	0
21	0x0000000F58AC6CD0	Service broker mappi...	CACHESTORE_BROK...	56	0	6	0
22	0x0000000F5604A080	View Definition Cache	CACHESTORE_VIEW...	16	0	0	0
23	0x0000000F5604ACD0	Notification Store	CACHESTORE_NOTIF	16	0	0	0

	cache_address	name	type	entry_address	entry_data_address	in_use_count	is_dirty	disk_ios_count	context_switches_count	original_cost	cu
1	0x0000000F5A342CD0	Object Plans	CACHESTORE_OBJCP	0x0000000F5A34E930	0x0000000F45A22150	1	0	0	25	2048	2
2	0x0000000F5A350080	SQL Plans	CACHESTORE_SQLCP	0x0000000F44D81D...	0x0000000F4346A150	2	0	0	0	0	0
3	0x0000000F5A350080	SQL Plans	CACHESTORE_SQLCP	0x0000000F44D81630	0x0000000F44D9C1...	1	0	0	11	0	0
4	0x0000000F5A350080	SQL Plans	CACHESTORE_SQLCP	0x0000000F44D80E...	0x0000000F45940150	1	0	0	12	0	0
5	0x0000000F5A350080	SQL Plans	CACHESTORE_SQLCP	0x0000000F44D80730	0x0000000F44DEC1...	1	0	0	0	1	1
6	0x0000000F5A350080	SQL Plans	CACHESTORE_SQLCP	0x0000000F44D80040	0x0000000F44D48150	1	0	0	15	0	0
7	0x0000000F5A350080	SQL Plans	CACHESTORE_SQLCP	0x0000000F4325F780	0x0000000F45D6C1...	1	0	0	0	1	1

Query executed successfully. IDEA-PC\SSQL (14.0 RTM) idea-PC\user (53) master 00:00:00 8,281 rows

These commands help us in check if memory is the bottleneck which is affecting the performance of the server. It is also helpful when we have to look at logs and system catalog tables.

### III. Understand the advantages and disadvantages of each RAID level

RAID (redundant array of independent disks) is a data storage virtualization technology that combines multiple physical disk drive components into one or more logical units for the purposes of data redundancy and performance improvement. Data is distributed across the drives in one of several ways, referred to as RAID levels, depending on the required level of redundancy and performance.

Raid levels	Advantages	Disadvantages
Raid level 0: Stripped disk array without fail tolerance	Complete utilization of storage capacity	A single drive failure will result in complete data loss
Raid level 1: mirroring and duplexing	Simple and easy to implement technology	Usable data storage capacity is only half of the total drive capacity
Raid level 2: hamming code ecc	High data transfer rates	Inefficient as very high ratio of ecc disks to data disks
Raid level 3: parallel transfer with parity	High read and write transfer rate	Difficult and resource intensive to use as a software rate
Raid level 4: independent data disks with shared parity disks	High aggregate read transfer rate	Worst write transaction rate
Raid level 5: independent data disks with distributed parity blocks	Highest read data transaction rate	Difficult to rebuild in an event of a disk failure
Raid level 6: independent data disks with two independent distributed parity schemes	Perfect solution for mission critical applications	More complex controller design
Raid level 10: very high reliability with high performance	Has higher performance	Very limited scalability at a very high inherent cost
Raid level 50: high i/o rates and data transfer performance	High data transfer rates	Very expensive to implement

## IV. Performance evaluation when tables are across multiple devices

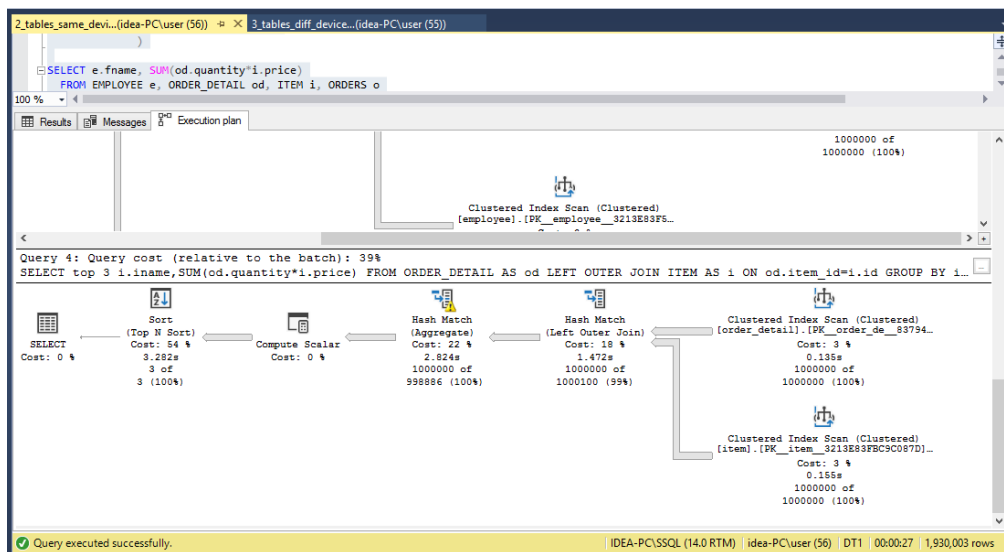
The tables of the database are spread across two devices (D and G drives). This is done so that the input-output gets divided across the disks and the queries are executed faster.

### Comparing:

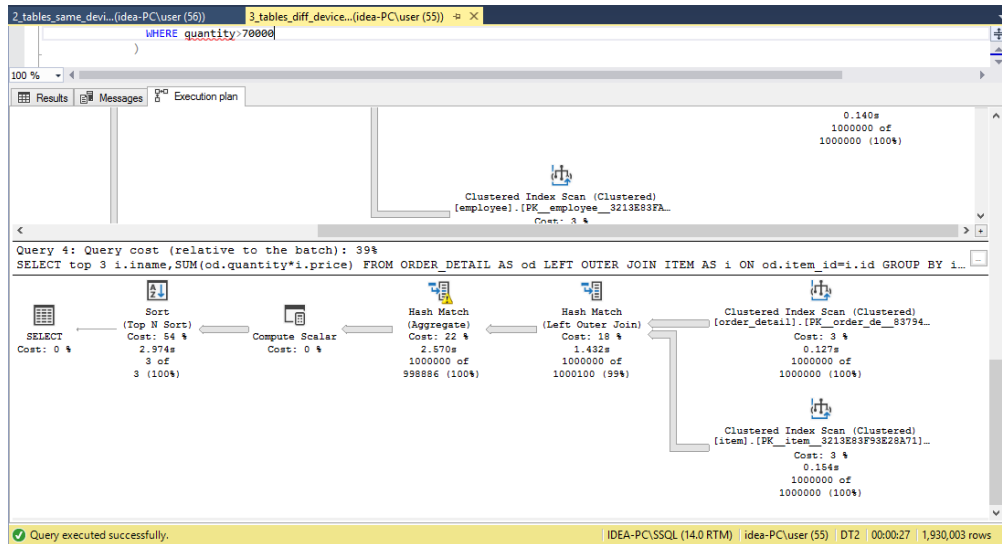
	Same device	Different device
Inserting data into the tables (1M records)	Execution Time: 00:03:18	Execution Time: 00:01:53
Query execution	Execution Time: 00:00:27	Execution Time: 00:00:27
Sort, hash match(aggregate, join)	Time taken was slightly more compared to when tables are on different device.	Time taken was slightly less compared to when tables are on same device.

### Inference:

The io time and the execution time is slightly reduced when tables are spread across multiple devices.





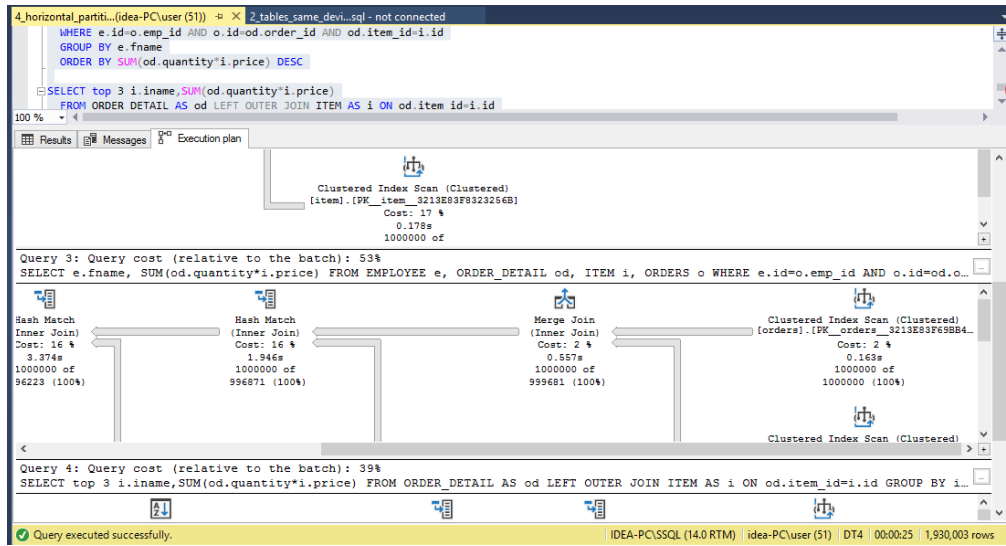


## V. Performance evaluation when a table is partitioned across multiple devices

Horizontal partitioning: Horizontal partitioning divides a table into multiple tables that contain the same number of columns, but fewer rows. As order\_detail table is assessed multiple times, order\_detail is partitioned using partition function and schema.

### Comparing:

	Not partitioned	Partitioned
<b>Inserting data into the tables (1M records)</b>	Execution time: 00:03:18	Execution time: 00:02:38
<b>Query execution</b>	Execution Time: 00:00:27	Execution Time: 00:00:25
<b>Queries on order_detail alone</b>	Execution Time: 6550ms	Execution Time: 3814ms
<b>Sort, hash match(aggregate, join)</b>	Time taken was slightly more compared to when table is partitioned.	Time taken was slightly less compared to when table is not partitioned.



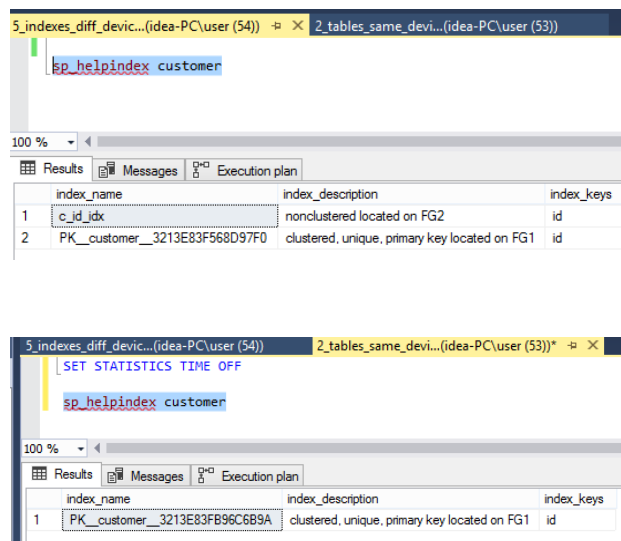
## Inference:

Although there is no much difference in the estimated io cost, but a significant reduction in time is noticed when a table is partitioned.

## VI. Performance evaluation on creation of indexes and storing them in separate devices

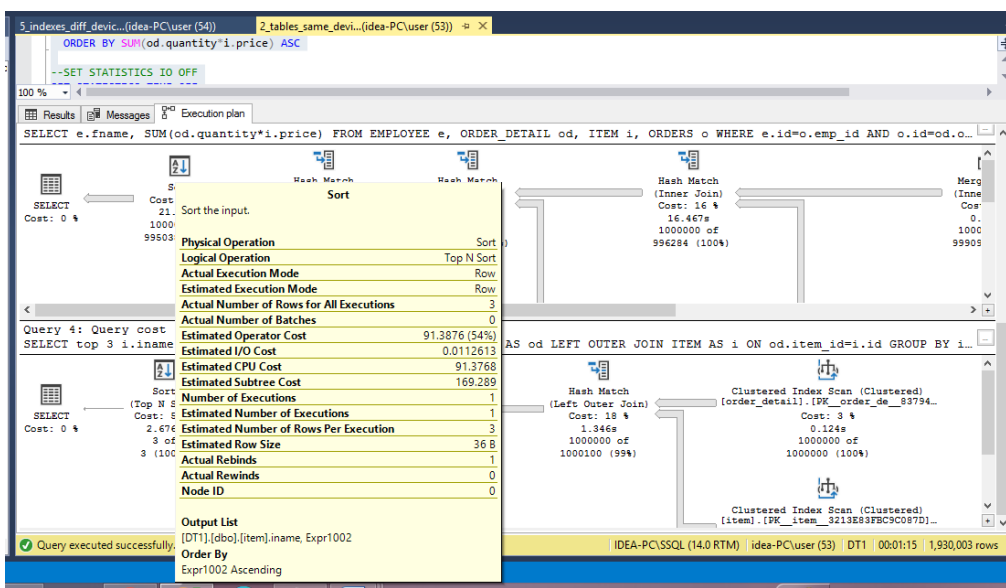
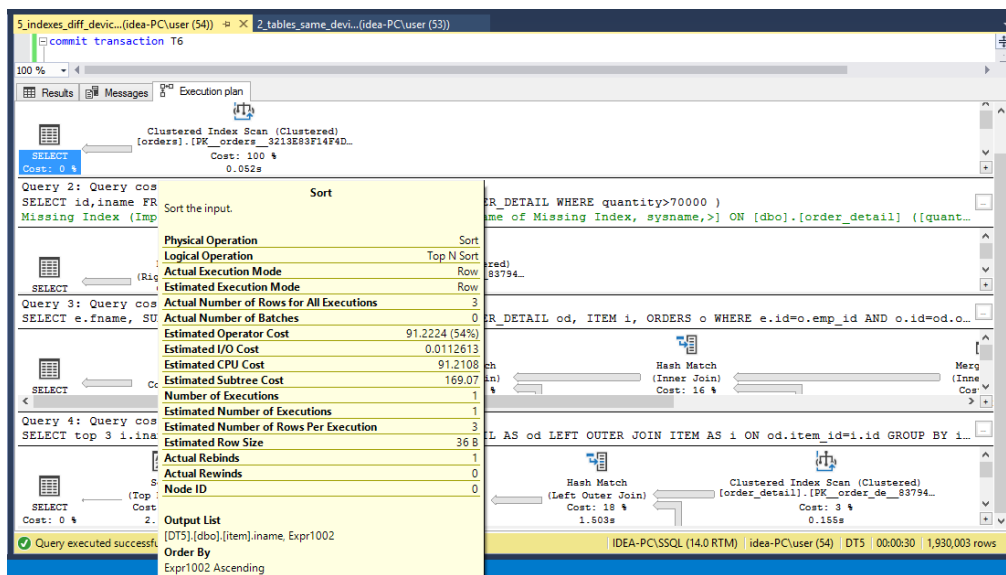
Clustered indexes were created on all the tables and were stored on a different device.

### Example:



## Comparing:

	clustered index(pk)	Non-clustered index (stored in different device)
Inserting data into the tables (1M records)	Execution time: 00:03:18	Execution time: 00:00:09
Query execution	Execution Time: 00:01:15	Execution Time: 00:00:30
Estimated i/o, cpu cost	Cost is higher compared to non-clustered index across multiple devices	Comparatively lower cost and faster execution

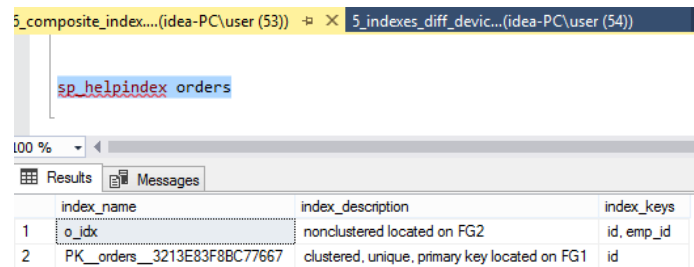


### Inference:

Creating non-clustered indexes across multiple devices not only helped in faster execution, but also affected the i/o and cpu cost by reducing it. Spreading over multiple disks reduces i/o (which is a potential bottleneck). It performs better when separated as there is no seek time and just rotational latency.

## VII. Performance evaluation on creation of composite indexes and storing them in separate devices

A composite index is an index on two or more columns of the table. Based on the queries, we find those attributes other than the primary key, which are mostly used and create a composite index on that.

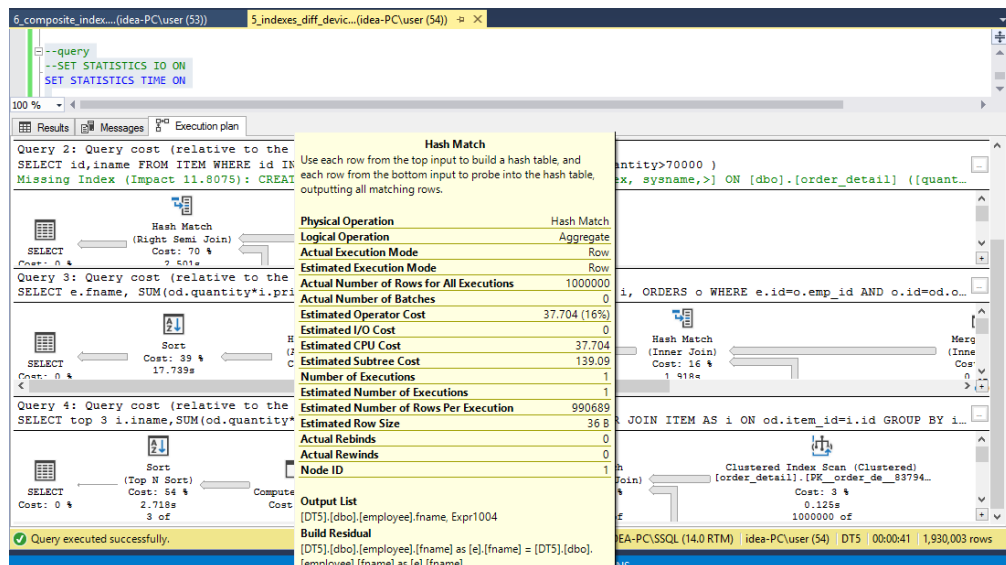
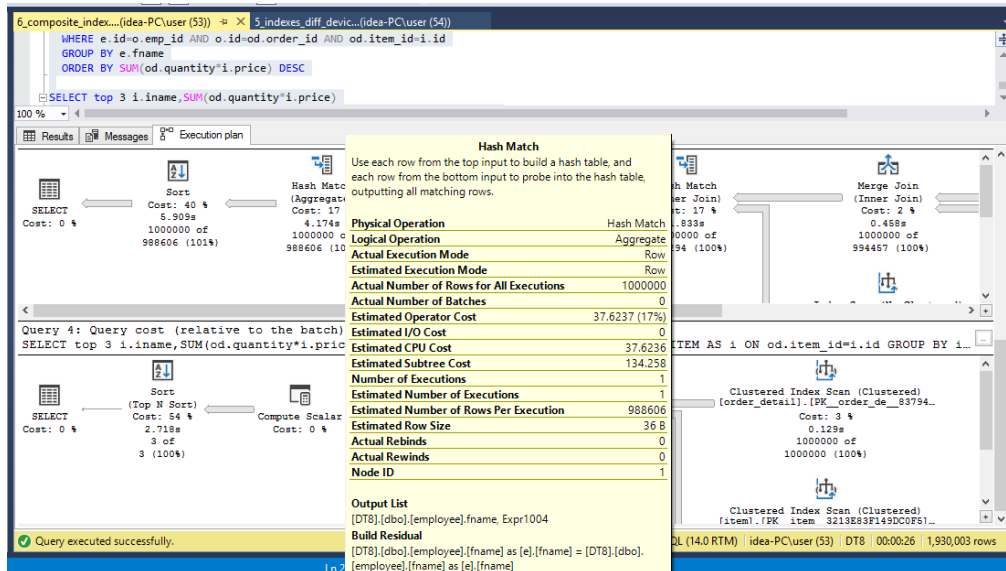


	index_name	index_description	index_keys
1	o_idx	nonclustered located on FG2	id, emp_id
2	PK_orders__3213E83F8BC77667	clustered, unique, primary key located on FG1	id

### Comparing:

	clustered index(pk)	Composite index (multiple columns)
Inserting data into the tables (1M records)	Execution time: 00:03:18	Execution time: 00:02:14
Query execution	Execution Time: 00:01:15	Execution Time: 00:00:26
Estimated i/o, cpu cost	Cost is higher compared to composite indexes across multiple devices	Comparatively lower cost and faster execution

## Execution:



## Inference:

Composite indexes has also helped in faster execution and reducing the cost.

**Creating indexes thus help improve the performance of queries.**