
# MASTER OF COMPUTER APPLICATIONS

## SEMESTER 1

# RELATIONAL DATABASE MANAGEMENT SYSTEM

# Unit 9

# Parallel Database Architectures for Parallel Databases

## Table of Contents


| SL No | Topic | Fig No / Table / Graph | SAQ / Activity | Page No |
|---|---|---|---|---|
| 1 | Introduction | - | - | 4 |
| | 1.1 Objectives | - | - | |
| 2 | Parallel Database | 1, 2, 3, 4 | 1, I | 5-11 |
| | 2.1 Advantages of parallel database | - | - | |
| | 2.2 Disadvantages of parallel database | - | - | |
| | 2.3 Parallelism in Database Management System | - | - | |
| 3 | Parallel Query Evaluation | 5 | 2 | 11-14 |
| | 3.1 Parallel query processing | - | - | |
| | 3.2 When to implement parallelism | - | - | |
| | 3.3 How parallel-execution works | - | - | |
| | 3.4 Parallelised SQL statements | - | - | |
| 4 | Parallelising Individual Operations | 6, 7, 8 | 3 | 14-17 |
| 5 | I/O Parallelism | 1 | 4 | 18-20 |
| | 5.1 Partitioning techniques (number of disks = n) | - | - | |
| | 5.2 Comparison of partitioning techniques | - | - | |
| 6 | Inter-Query Parallelism | 9 | 5 | 21-22 |
| 7 | Intra Query Parallelism | 10, 11, 12, 13 | 6 | 22-24 |
| | 7.1 Intra partition parallelism | - | - | |
| | 7.2 Inter partition parallelism | - | - | |

| | | | | |
|---|---|---|---|---|
| 8 | Inter Operation and Intra Operation Parallelism | - | 7, II | 25 |
| 9 | Design of Parallel Systems | - | 8 | 26 |
| 10 | Summary | - | - | 27-28 |
| 11 | Glossary | - | - | 29 |
| 12 | Terminal Questions | - | - | 29 |
| 13 | Answers | - | - | 30 |
| 14 | References | - | - | 30 |

# 1. INTRODUCTION

In the previous unit, you studied Concurrency Control and its various related aspects such as enforcing, serialisability by locks, locking systems, architecture for a locking scheduler, managing hierarchies of database elements, concurrency control and database recovery management. In this unit, we will introduce you to transaction processing.

Databases are becoming increasingly large as they contain large volumes of transaction data and various types of multimedia objects such as images. Hence, large-scale parallel database systems are gradually used more and more for storing large volumes of data; process the time-consuming decision-support queries and providing high throughput for transaction processing.

Thus, in this unit we will look at the issue of parallelism and data distribution in a DBMS. You will learn about the basic concepts of parallel database and alternatives for parallel database architecture. Thereafter, you will be introduced to the concept of data partitioning and study its influence on parallel query evaluation. This unit explores a variety of parallelisation techniques, including I/O parallelism, inter-query and intra-query parallelism, and inter operation and intra operation parallelism. The unit will wind up with the various parallel-system design issues.

## 1.1 Objectives

*After studying this unit, you should be able to:*

- *Describe parallel databases and its architecture*
- *Identify parallel query evaluation*
- *Demonstrate parallelisation of individual operations in parallel databases*
- *Identify the concept of I/O parallelism*
- *Differentiate between inter-parallelism and intra-parallelism*
- *Compare intra operation with inter operation parallelism*
- *Recognise the various design issues of parallel system*

## 2. PARALLEL DATABASE

A parallel database can be defined as a database that run multiple instances to "share" a single physical database. A variety of hardware architectures allow multiple computers to share access to data, software, or peripheral devices. A parallel database is designed to take advantage of such architectures.

Parallel database systems are based on the concept of "parallelism in data management". In order to deliver high-performance and high-availability, database servers are placed at a much lower price than equivalent mainframe computers.

Parallel database are widely used now-a-days because of its unique benefits. The technology of Parallel database helps to benefit specific types of applications by enabling features such as higher performance, greater flexibility, high availability and capacity to serve many users simultaneously. You might wonder how a parallel database is able to do all this efficiently.

In a parallel database technology, more than one CPU is accessible to an application, hence higher speed up and scale up can be achieved. Also the nodes are separated from each other; therefore any malfunction at one node does not affect the entire system. In case of any failure, one of the properly working nodes covers for the failed node and the system carries onto furnish data access to users. And, this provides high database availability.

In parallel database, allocation and de-allocation of instances can also be done as per necessity. For example, more instances can be allocated if there is increase in database demand and also some instances can be de-allocated if the demand is less. Parallel database also makes it possible to get over the memory constraint by empowering a single system to serve multitude of users.

### 2.1 Advantages of Parallel Database

There are numerous advantages of parallel database technology. Some of these have been listed below:

1. Increased throughput
2. Decreased response time
3. Ability to process an extremely large number of transactions

4. Substantial performance improvement
5. Increased availability of system
6. Greater flexibility
7. Possible to serve large number of users

## 2.2 Disadvantages of Parallel Database

Along with advantages it also carries some disadvantages as listed below:

1. More start-up costs as when several processes start in parallel they tend to easily dominate the real computation time.
2. Interference problem is created due to the slow-down which each new process imposes on the other process.
3. The service time of the slowest step of task is actually the service time for the system.

## 2.3 Parallelism in Database Management System

Parallelism in Database Management System is available in various forms. It also has numerous goals and objectives to fulfil. All this and a detailed explanation of the parallel DBMS architecture will be discussed in the following sections:

**Goals/Objectives of parallelism in database**

There are mainly two objectives of parallelism in database:

1. ***Speed Up:*** First objective is to speed up the processing of a given task/query i.e. to reduce the response time. This is done with the help of additional hardware that helps to process the same task in lesser time as compared to a single hardware. It is as simple as two men performing a task will take lesser time as compared to one man doing it alone.

   Speed up is measured by the formula:

   Speed up = Time _ Original / Time _ Parallel

   If an original system takes 2 minutes to process a task and a parallel system takes 1 min to process the same task, then the speed up will be 2 (i.e. 2min/1min).

2. ***Scale Up:*** The next goal of parallel database is scale up. Scale up means to process a larger number of jobs in the same time period. It may also be said to increase the throughput of the system.

Scale up can be calculated by applying the below given formula:

Scale up = Volume _Parallel/ Volume _ Original

For example: If an original system processes 1000 transactions in a minute and a parallel system processes 2000 transactions in a minute then the scale up will be 2 (i.e. 2000/1000).

Most of the traditional DBMS used relational database approach, which was dominating at onetime - it is still a dominating database approach. The fact that relational queries are ideally suitable for parallel-execution, gave great interest to many researchers in adopting parallelism in DBMS, and resulted in giving birth to parallel DBMSs. Relational queries is composed of uniform operations implemented on consistent streams of data. Every operator creates one new relation; therefore the operators can be compiled into parallel data-flow graphs.

**Forms of parallelism**

There are two forms of parallelism that can be applied to DBMSs.

1. ***Pipelined parallelism:*** In a pipelined parallelism approach, one operation's output is streamed into the input of some other operator and these two operators can function simultaneously in series. Figure 9.1 depicts an example of pipelined parallelism in which the output of scan is immediately fed into the input of sort where sort is executing in parallel to the data scan.
2. ***Partitioned parallelism:*** In a partitioned parallelism approach, the input is partitioned between different processors & memories, and also the operator can often be divided into several autonomous operators each working on a part of the data. Figure 9.1 illustrates an example of partitioned parallelism in which four processors scan and sort the input simultaneously and the results of all four are merged together to generate the final output.

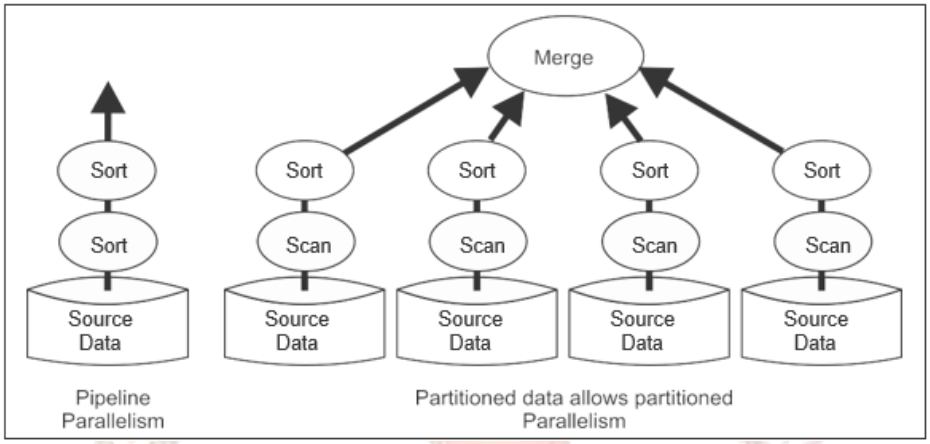


**Fig 9.1**: Pipelined and Partitioned Parallelism

**Parallel DBMSs architecture**

Now we will briefly discuss the parallel DBMS architecture. There are mainly three machine architectures available on which you can run parallel DBMSs so as to minimise the response time and maximise the throughput.

These three architectures are discussed below:

(a) **Shared memory multiprocessor**: In a shared-memory multiprocessor architecture, several processors share storage disks and common memory with assistance of interconnected network. Figure 9.2 shows such one shared memory architecture.

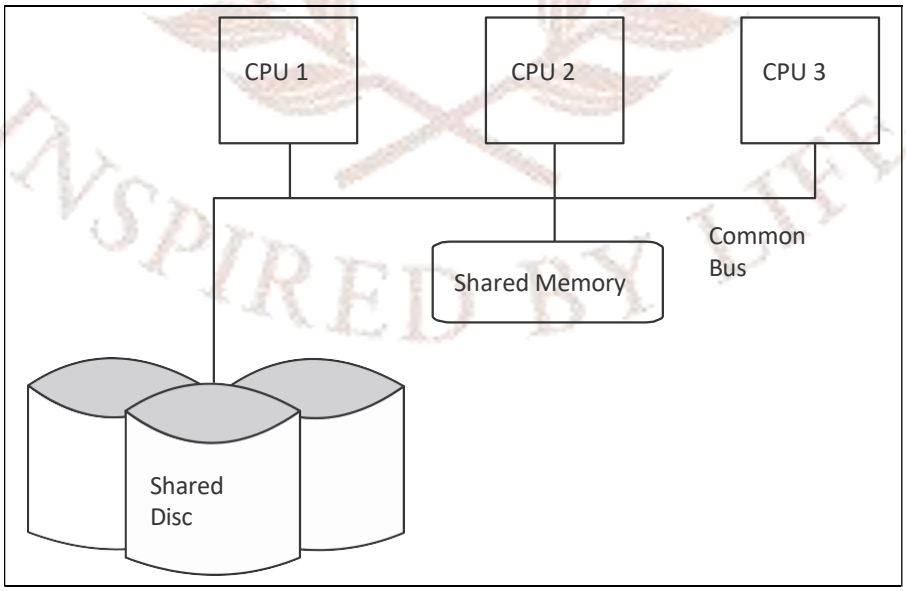


**Fig 9.2:** Shared Memory Architecture

This shared-memory architecture is suited for lower degree of parallelism i.e. where there are not many processors. But for this type of system, one major problem is network interference. Moreover, partitioning creates many of the skew and load balancing problems faced by shared nothing machine.

(b) ***Shared nothing multiprocessor:*** In a shared-nothing architecture, each processor has its own personal memory besides one or more disk storage. As Shared-nothing multiprocessor runs only question &answer through the network, low traffic is likely in the inter-connection network. This gives low network intervention among processors, and hence permits high scalability. Figure 9.3 shows a shared nothing illustration.

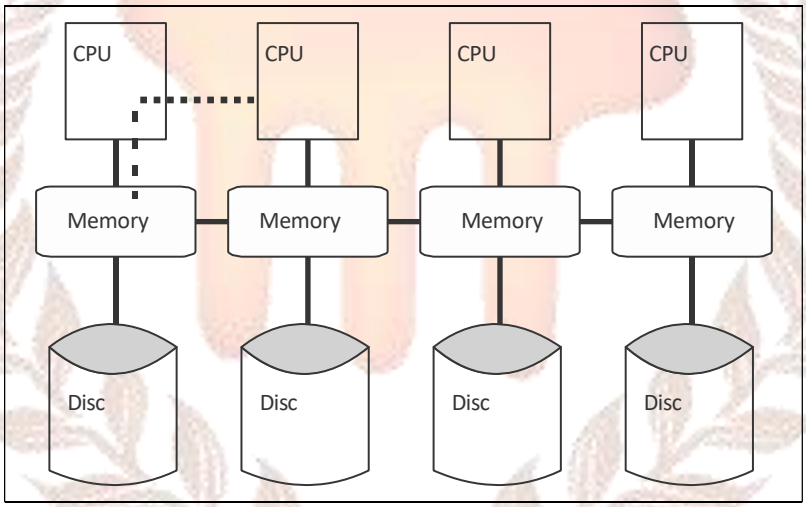


**Fig 9.3:** Shared-Nothing Architecture

Shared-nothing parallel machine architecture is possibly the most efficient architecture upon which parallel database system should be carried out.

(c) ***Shared-disk multiprocessor:*** In shared-disk architecture, every processor has its own personal memory and only shares storage disks via an inter-connection network. Shared-disk multiprocessor aids in improving network interference difficulty of shared -memory multiprocessor. Figure 9.4 shows a shared -disk illustration.

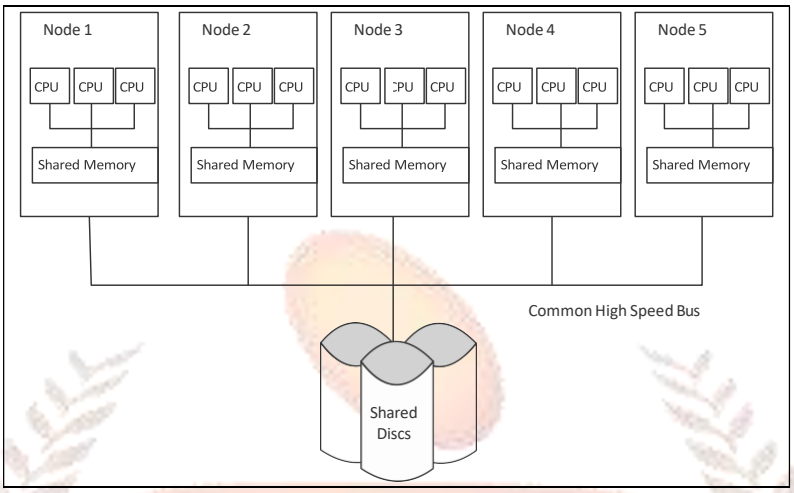


**Fig 9.4:** Shared-Disk Architecture

This architecture is effective for following situations:

- big read-only database
- database in which there is no simultaneous sharing

It is not really suitable for database applications that reads & writes shared database. When a processor wants a shared disk page for write concurrency control, it becomes an issue. To implement this in shared-disk architecture inter-processor communication is necessary for reservation and release. There are various optimisations of such protocol, but they all exchange large data pages and reservation messages and. It produces processor interference besides producing delays in heavy traffic on the shared up inter-connection network.

**SELF-ASSESSMENT QUESTIONS – 1**

1. Nearly all traditional DBMS uses relational database approach. (True/False)
2. In which architecture, each processor has private memory and one or more private disk storage?
   a) Shared memory
   b) Shared Disk
   c) Shared nothing
   d) Pipelined

**Activity 1**

Briefly compare the three alternative design approaches for parallel databases based on their potential advantages and disadvantages.

## 3. PARALLEL QUERY EVALUATION

Optimisation of Query evaluation is done by using parallel optimisation methodologies for making repartitioning more competent. Efficiency is enhanced by identifying the potential partitioning necessities for obtaining, parallelism for query operation, and by identifying when the data's partitioning property fulfils the partitioning needs of a query operation.

A database management system in agreement with the invention uses parallel query processing approaches to optimise repartitioning of data, or to void it altogether. Now let us understand the basics of parallel query processing.

### 3.1 Parallel Query Processing

In parallel query processing, multiple processes together at the same time to handle one individual SQL statement. The Database server can handle the statement more swiftly in comparison to a single server by segregating the work necessary to process a statement among various multiple servers.

The feature helps to dramatically enhance functioning for data intensive operations related with decision-support applications and even extremely big database environments. Parallel query feature is most useful for SMP (Symmetric multiprocessing), clustered or MPP (massively parallel systems) because of the fact that in such types of systems the query processing could be efficiently split up among various central processing units on one single system.

In parallel query processing, the query is parallelised dynamically at the execution time itself. It automatically adapts to optimise itself for parallelisation, if the location or distribution of the data changes.

## 3.2 When to Implement Parallelism

Parallel-execution is helpful for various types of operations that access considerable amounts of data. Parallelism enhances performance for:

- Queries
- formation of large indexes
- volume inserts, updates, and deletes
- Aggregations and copying

Parallel-execution is advantageous to systems with the following characteristics:

- Symmetric multiprocessors (SMP), clusters, or massively parallel systems (for example, multiple CPUs)
- Sufficient I/O bandwidth
- Under-utilised CPUs or occasionally used Central Processing Units (for instance, systems where CPU use is characteristically less than 30percent)
- Adequate memory to sustain extra memory intensive processes like hashing, sorts and I/O buffers

## 3.3 How Parallel-Execution Works

The basic unit of work in parallelism is called a **granule**. When a database instance starts up, the database makes a pool of parallel-execution servers that are present for any parallel operation. When performing a parallel operation, the coordinator acquires parallel-execution servers from the group and allocates them to the operation. The One process, by the name of **parallel-execution coordinator**, assigns execution of one granule to many parallel-execution servers and co-ordinates results from every server processes and return results to the user.

Figure 9.5 below displays many such servers executing a scan of the table employees.

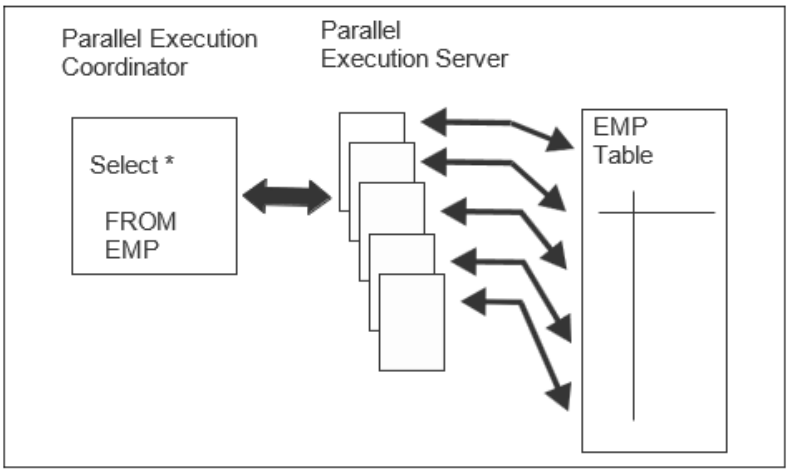


**Fig 9.5**: Parallel Full Table Scan

As you can see in the Figure 9.5, the table EMP is segmented dynamically into various granules. This task of granule generation is done by coordinator. This granule is nothing else but an assortment of blocks of table employees. Now each granule is read by a distinct parallel-execution server. Note that the mapping is not static, but is decided at the time of execution. When one execution server completes studying rows in the table employees analogous to a granule, it retrieves additional granule from coordinator if some granules are left behind.

This process continues until all the granules are being used up, in other words the complete table employee has been successfully read. After this, the servers dispatch results back to the coordinator which completes the task of re-assembling pieces into the needed full scan.

The total servers allocated to one single operation are the DOP (degree of parallelism) for the operation. Various operations among the identical SQL statement have the same scale of parallelism.

## 3.4 Parallelised SQL statements

Every SQL statement goes through an optimisation and parallelisation process where the parsing is done. Optimisation is done by the Optimiser. A database is capable of adapting itself to the new situation if it finds a more optimum execution plan for a given SQL statement.

After the optimiser decides execution plan for a given SQL statement, thereafter the coordinator decides the appropriate parallelisation method for every operation in execution plan.

Therefore it is the responsibility of the coordinator to determine if an operation must be executed in parallel and if so then how many servers must be required. The number of servers employed for an operation is 'degree of parallelism'.

**SELF-ASSESSMENT QUESTIONS – 2**

3. Oracle is capable of increasing or decreasing the number of parallel- execution servers based on the requirement. (True/False)
4. Basic unit of work in parallelism is _____.

## 4. PARALLELISING INDIVIDUAL OPERATIONS

In this section, you will learn how individual operations are parallelised in a database. You have learnt in previous section that before enlisting query server processes, the query coordinator process scrutinises the operations in the query execution plan to determine whether the individual operations can be parallelised.

The Server can parallelise the following operations:

- sorts
- joins
- table scans
- table population
- index creation

**Partitioning rows to each query server**

An essential step in parallelisation is partitioning the rows to each query server. This task is done by the query server. The process (query- coordinator process) determines the partitioning needs of each and every operation. Partitioning requirement of an operation is the method in which rows acted-on by the operation must be partitioned, or divided between the query server processes. The partitioning maybe any of the given below:

- random
- round robin
- hash
- range

***Note:*** You will learn in detail about the partitioning techniques in next section.

The next step involves determining the ordering requirement for every operation in execution plan. This is again done by the query coordinator. The coordinator establishes the flow of data of the statement i.e. which operation will be succeeded and preceded by which operation. Operations that need the output of other operations are known as parent operations.

**Parallelism between operations**

After deciding upon the partitioning and ordering, the database can proceed on to parallelism between operations. For doing so it is essential that the child operations have been executed before the parent operations, as parent operation requires the output of child operations to consume.

For example consider a query and its data flow along with data flow diagram as given in Figure 9.6 and 9.7 respectively:

```
SELECT  dname,  MIN(Age),  AVG(Age)
          FROM  emp,  dept
            WHERE emp.deptno =  dept.deptno
            GROUP BY dname;
```

**Fig 9.6:** The Data Flow for the Above Query

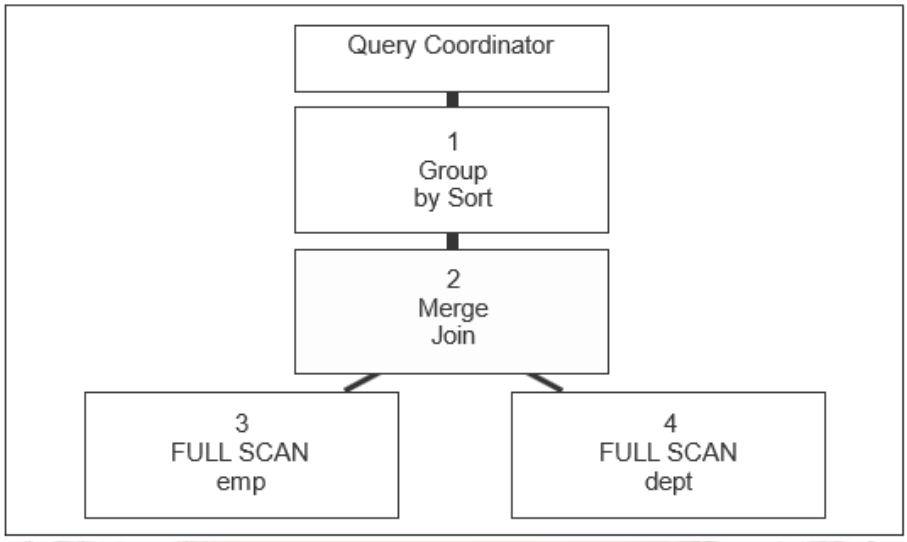


**Fig 9.7:** Data Flow Diagram

Figure 9.8 below illustrates the parallel-execution of our sample query.

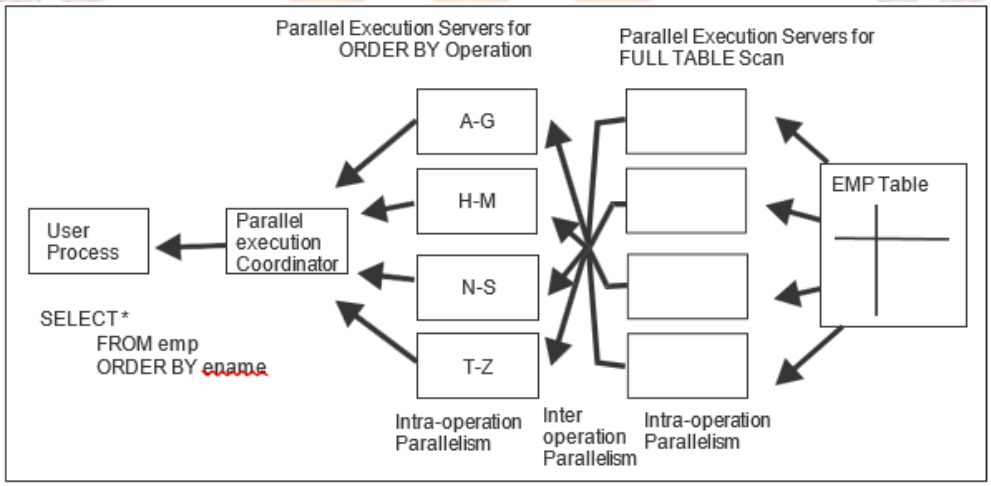


**Fig 9.8:** Inter operation Parallelism and Dynamic Partitioning

As you see in Figure 9.8 that while the query servers are generating rows in the FULL SCAN DEPT operation, another set of query servers can start performing the MERGE JOIN operation to consume the rows. When the FULL SCAN DEPT operation is complete, the FULL SCAN EMP operation can begin to produce rows.

It is noteworthy that in reality there are 8 parallel-execution servers working in the above given query, though the degree of parallelism is 4. The reason being, a parent & child

operator can be executed simultaneously on account of the process of interoperation parallelism.

You should also notice that all of the servers occupied in the scan operation transmit rows to the suitable server executing the sorting operation. If a row read by a parallel-execution server includes one value for the 'ename' column between A & G, that row is sent to the 1st order by parallel- execution server. After the completion of scan operation, the sorting processes could send back the sorted results to coordinator, which returns the query results to the users.

**Degree of parallelism**

The parallel-execution coordinator can enrol 2 or more parallel-execution servers of the instance to treat a given statement. The total quantity of parallel-execution servers that are linked with a single operation is known as the ***'degree of parallelism'.*** It refers directly to intra operation parallelism only. If inter operation parallelism is achievable, then the total number of parallel-execution servers for one statement can be double the stipulated degree of parallelism. Therefore, only up to two sets of parallel-execution servers can run at one time. Every group of parallel-execution servers could process numerous operations. To ensure optimal inter operation parallelism, only 2 sets of parallel-execution servers should be active.

**SELF-ASSESSMENT QUESTIONS – 3**

5. Which of the following operations require the output of other operations?
   a) Child operations
   b) Dependent operations
   c) parent operations
   d) Inter-related operations
6. The number of _____ linked with a single operation is known as the degree of parallelism.

## 5. I/O PARALLELISM

I/O parallelism (Input/output parallelism) is the simplest type of parallelism. This parallelism attempts to minimise the time required to retrieve relations from disk by **partitioning** the relations on multiple disks. In this the input data is partitioned and thereafter each partition is processed in parallel. The results obtained are then combined after the processing of all partitioned data. This technique greatly reduces the retrieval time. I/O parallelism is also referred to as ***data partitioning.***

In horizontal partitioning, the tuples of a relation are distributed between several disks in a manner such that each tuple is on one disk.

Partitioning features can help to significantly enhance data access and improve overall application performance. Employing the partitioning techniques explained here, you could tune SQL statements to avert avoidable index & table scans (by employing partition pruning).

You can also enhance the execution of very big join operations when huge amounts of data (for instance, millions of rows) are connected jointly by employing partition wise joins. Ultimately, partitioning data significantly increases managing capability of big databases and significantly cut down the time period needed for administrative tasks like restore and backup.

### 5.1 Partitioning Techniques (number of disks = n)

There are mainly three types of partitioning techniques i.e., Round robin, Hash partitioning and Range partitioning. Each partitioning method has different advantages and design considerations. Thus, each method is more appropriate for a particular situation. Now we will discuss these techniques in detail.

(a) ***Round robin:*** The most simple partitioning approach divides tuples between the fragments in the manner of round robin tournament. In this partitioning technique, the mth tuple of a Relation Z is inserted to disk dm mod n. In simple words, the disk takes turns while receiving new tuples. Therefore in this method tuple S would be placed on D1, tuple T would be placed on D2, tuple U on D3 & so on.

Round robin approach is the partitioned variant of the standard entry- sequence file.

***Advantage:*** Round robin partitioning is best when all the applications want to access relation by scanning sequentially all of it on each and every query.

***Disadvantage:*** Round robin technique is not suitable for sophisticated access of the relation. For example in case of Join, this technique would take too much of time.

(b) ***Hash partitioning:*** In this technique we select attributes (one or more) as partitioning attributes. We select hash function l with range 0 n .

Each tuple or row of the primary relation is hashed on the specific attribute. The output for the hash function leads to a data that is transferred to the disk m. Let m refer to the result of hash function l applied to the partitioning attribute value of a tuple. Send tuple to disk m.

***Advantages:*** Hash Partitioning helps to avoid data skew i.e. data is evenly distributed across the disk. Hash partitioning is practically suitable for the applications which want only associative and sequential access to data. Tuples are positioned by applying a hashing function to any one attribute of the each tuple.

This function indicates the allotment of position of tuple on one specific disk. Associative access to the tuples with one particular attribute value could be directed to one single disk, bypassing the extra work of starting queries on numerous disks.

***Disadvantages:*** It is not suitable for point queries (queries that involve exact matches) on non-partitioning attributes. For example:

```
SELECT *
FROM STUDENT
Where Stud_Age>8 AND Stud_Age<17;
```

This type of range queries would require more time if implemented with Hash partitioning.

(c) ***Range partitioning:*** In this technique, the administrator chooses an attribute as the partitioning attribute and specifies attribute-values within a certain range to be placed

on certain disk. Let n be the partitioning attribute value of a tuple. Tuples such that *ni ≥ ni+1* go to disk a + 1. Tuples with *n<n0* go to disk 1 and tuples with *n ≥ nn-2* go to disk a1.

Thus, range partitioning helps to distribute a contiguous attribute value ranges to each disk.

***Advantages:*** Range partitioning bundles tuples with same types of attributes collectively in the similar partition. It is effective for grouping data and also sequential & associative access.

Range partitioning is best suited for performing range based queries and is also good for point queries (finding exact matches) involving a partitioning attribute.

***Disadvantages:*** The basic disadvantage with this partitioning is that, it may result in data skew and execution skew. In data skew, all the data is located in one partition. In execution skew, all executions take place in one partition. On the contrary, round-robin and hashing are less prone to such type of skew problems.

## 5.2 Comparison of Partitioning Techniques

Table 9.1 below shows the comparison between the three partitioning techniques on the basis of sequential scan, point query and range query execution.

**Table 9.1:** Comparison between Round robin, Hashing and Range techniques

| | Round Robin | Hashing | Range |
|---|---|---|---|
| Sequential scan | Best/Good parallelism | Good | Good |
| Point query | Difficult | Good for hash key | Good for range vector |
| Range Query | Difficult | Difficult | Good for range vector |

**SELF-ASSESSMENT QUESTIONS – 4**

7. Partitioning data very much improves managing capacity of very big databases. (True/False)
8. _____ partitioning is very suitable for point queries and also range queries.

## 6. INTER-QUERY PARALLELISM

Inter-query parallelism means executing different independent queries simultaneously on separate CPUs. Figure 9.9 below shows how three independent queries can be executed simultaneously by three different processors. Each request (task) runs on a single thread and executes on a single processor.

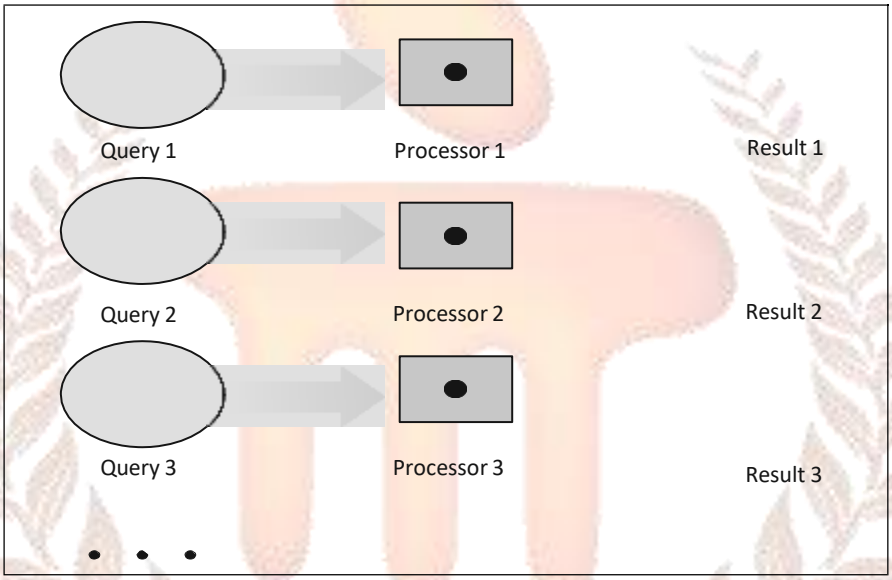


**Fig 9.9:** Inter-Query Parallelism

In inter-query parallelism the transactions or queries parallelly execute with each other. Inter-query parallelism is used effectively in OLTP (On-line transaction processing) applications. Hence it is used mainly to update a transaction processing system in order to maintain a greater number of transactions frequencies.

***Advantages:*** It is perhaps the easiest type of parallelism to maintain in a database system mainly in a shared memory system. Helps to increase transaction throughput. Helps OLTP applications to support more number of users simultaneously.

***Disadvantages:*** Response times of individual transaction/queries are not much faster as compared to the transaction or queries that run in isolation. It is quite complicated in a shared-nothing or shared-disk architecture.

**SELF-ASSESSMENT QUESTIONS – 5**

9. The full form of OLTP is _____.
10. Inter query parallelism aids in increasing the transaction throughput. (True/False)

## 7. INTRA QUERY PARALLELISM

In Intra query parallelism a single large query is broken into a number of pieces (subtasks) and those subtasks are executed in parallel on multiple processors. It is sometimes also referred as *parallel query processing*. Figure 9.10 below shows inter-query parallelism in which a large query is broken down into two subtasks 1 and 2 which are simultaneously executed on two different processors. The two results R1 and R2 combine together to form query results.

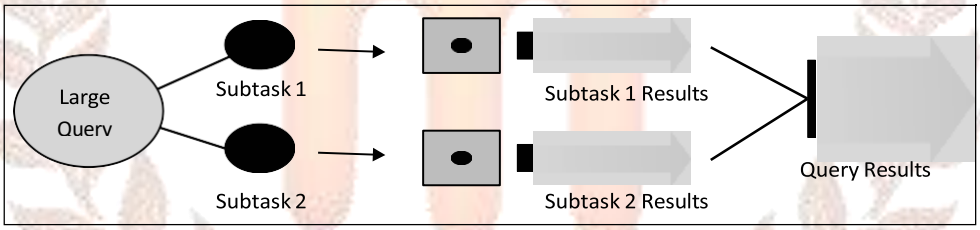


**Fig 9.10:** Intra-Query Parallelism

Now let us discuss in detail about intra partition parallelism and inter partition parallelism.

### 7.1 Intra Partition Parallelism

Intra partition parallelism relates to the capacity to break up a query into several parts within a single database partition and execute these parts at the same time. Intra partition parallelism splits up a particular database operation, such as database load, index creation, and SQL queries into numerous parts, all of which or many of them can be parallelly executed in one database partition. Intra partition parallelism can be applied to get advantage of multiple processors of a SMP (symmetric multiprocessor) server.

Figure 9.11 shows a query that is subdivided into 4 pieces that could be parallelly executed, each of them working with one single subset of the data. When this happens, the results can be sent back more quickly in comparison to query when it was running serially.

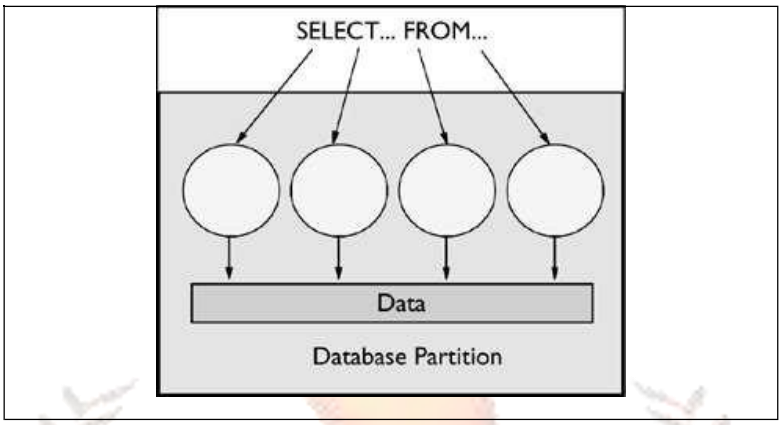


**Fig 9.11:** Intra Partition Parallelism

## 7.2 Inter Partition Parallelism

Inter partition parallelism relates to the capacity to break up a query into various parts across various partitions of a partitioned database which may be on one single machine or several machines. The query is run in parallel. Figure 9.12 shows a query that is subdivided into four parts that can be run in parallel. In this scenario, the results are returned more swiftly as compared to the query, if it were run in serial fashion on one single partition.

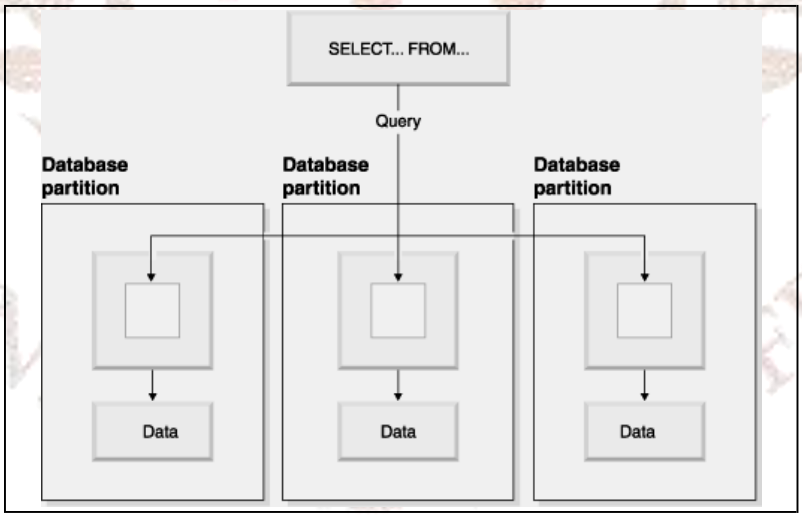


**Fig 9.12:** Inter-Partition Parallelism

One can also use intra partition parallelism and inter partition parallelism simultaneously. This unique grouping provides two attributes of parallelism, resulting in an even more impressive increase in the speed of processing of queries. Figure 9.13 shows a query in which

both intra partition parallelism and inter partition parallelism is implemented at the same time.

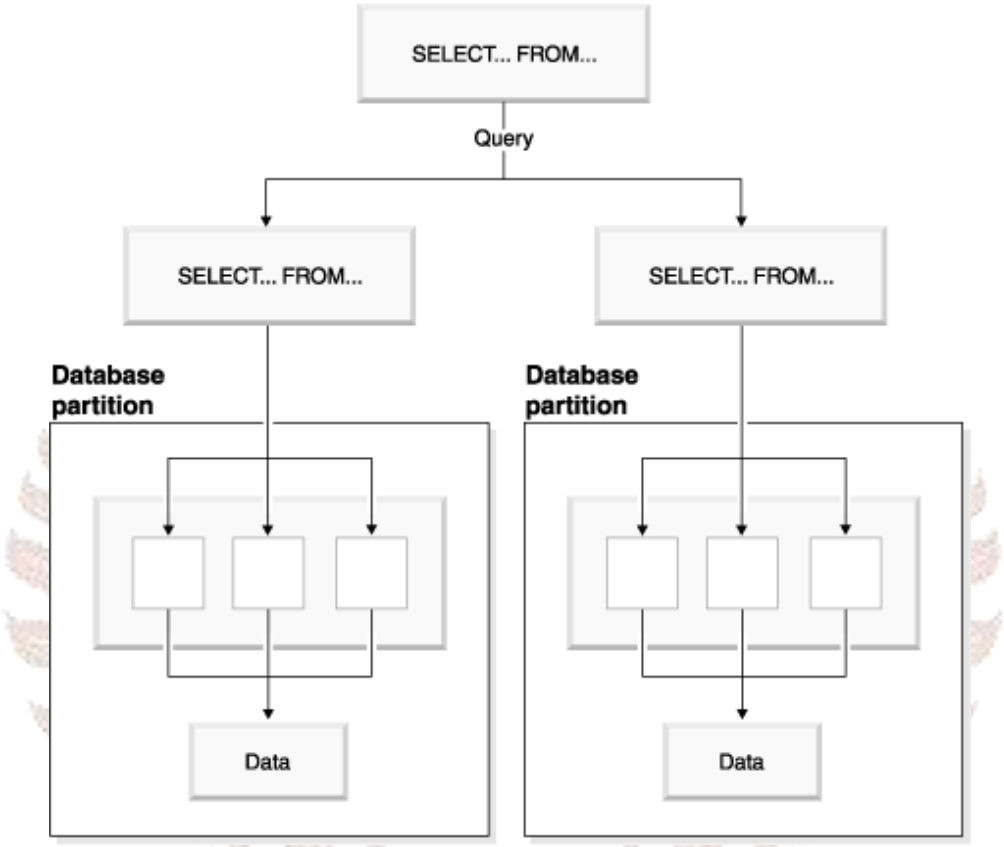


**Fig 9.13:** Simultaneous Implementation of Intra-Partition and Inter-Partition Parallelism

**SELF-ASSESSMENT QUESTIONS – 6**

11. Intra query parallelism is sometimes also referred as _____.
12. One cannot employ intra partition parallelism and inter partition parallelism at the same time. (True/False)

## 8. INTER OPERATION AND INTRA OPERATION PARALLELISM

Two complementary forms of intra query parallelism are interoperation parallelism and intra operation parallelism. In this section, we will discuss briefly about these two. We have already discussed about these two concepts in section 9.4.

***Intra operation Parallelism:*** In intra operation parallelism, the implementation of each single operation in query is parallelised. These operations may be of sort, join, projection etc.

As you might have observed that the number of operation are small as compared to the number of tuples/rows/records that needs to be treated by each operation, hence this performs better with increasing parallelism.

***Inter operation Parallelism:*** In case of inter operation parallelism, the various operations in a query expression are implemented in parallel.

**SELF-ASSESSMENT QUESTIONS – 7**

13. Intra operation parallelism performs better with increasing parallelism. (True/False)
14. In _____ parallelism, the various operations in a query expression are parallelly executed.

### Activity 2

Explore some of the practical applications of inter and intra operation parallelism.

## 9. DESIGN OF PARALLEL SYSTEMS

Designing parallel systems is not an easy task. There exist many issues associated to it. Some major issues in parallel systems designs are discussed below:

- Parallel processing of data from outside sources is required in order to manage large quantities of arriving data.
- Proper technique should be there to manage problem of data skew.
- Adaptability in case of failure of some disks or processors:
  - ❖ The chances of any processor or disk failure are more in the parallel system. The breakdown of any of the processor or single disk will lead to problems in entire system.
  - ❖ Operations even in case of degraded performance should be able to execute in spite of breakdown or failure.
- The system must support schema changes and online data reorganisation, like in the case of construction of index for terabyte databases could take hours, days or even weeks on a parallel system. Hence there is requirement for allowing other processes (insertions/updates/deletions) to be performed on relation even as index is being constructed.
- There should be support for schema changes and online repartitioning (concurrent processing).

**SELF-ASSESSMENT QUESTIONS – 8**

15. The chances of any processor or disk failing is more in a parallel system. (True/False)
16. There must be a suitable technique to manage _____ skew problem in parallel system design.

## 10. SUMMARY

Let us recapitulate the important points of this unit:

- A parallel database can be defined as the database having many memory areas that share one disk drive.
- Parallel database systems are based on the concept of "parallelism in data management".
- In a parallel database technology, more than one CPU is accessible to an application, hence higher speed up and scale up can be achieved.
- With Parallel database, we can make it possible to get over the memory constraint by empowering a single system to many users.
- Parallelism in Database Management System is available in various forms.
- There are two forms of parallelism that can be applied to DBMSs: Pipelined parallelism and Partitioned parallelism
- There are mainly three machine architectures available on which you can run parallel DBMSs; Shared memory multiprocessor, shared nothing multiprocessor and Shared-disk multiprocessor.
- Optimisation of Query evaluation is done by using parallel optimisation methodologies for making repartitioning more competent.
- In parallel query processing, multiple processes together at the same time to handle one individual SQL statement.
- In parallel query processing, the query is parallelised dynamically at the execution time itself.
- Every SQL statement goes through an optimisation and parallelisation process where the parsing is done.
- An essential step in parallelisation is partitioning the rows to each query server.
- The total quantity of parallel-execution servers that are linked with a single operation is known as the 'degree of parallelism'.
- I/O parallelism (Input/output parallelism) is the simplest type of parallelism.
- I/O parallelism is also known as data partitioning.

- There are mainly three types of partitioning techniques; Round robin, Hash partitioning and Range partitioning.
- Inter-query parallelism means executing different independent queries simultaneously on separate CPUs.
- In Intra query parallelism a single large query is broken into a number of pieces (subtasks) and those subtasks are executed in parallel on multiple processors.
- Intra partition parallelism relates to the capacity to break up a query into several parts within a single database partition and execute these parts at the same time.
- Inter partition parallelism relates to the capacity to break up a query into various parts across various partitions of a partitioned database which may be on one single machine or several machines.

## 11. GLOSSARY

- ***Degree of parallelism:*** The total number of parallel-execution servers for a single operation is known as 'degree of parallelism'.
- ***Intra-query parallelism:*** The ability to subdivide a single query into a number of parts and replicate them concurrently using either intra partition parallelism or inter partition parallelism or both.
- ***Parallel database systems:*** The database having various memory regions and sharing one single disk drive.
- ***Parallel I/O:*** The process of reading from or writing to two or more I/O devices concurrently.
- ***Shared disk architecture:*** In this model, every processor has its own memory and share storage disks only, through interconnection network.
- ***Shared memory multiprocessor architecture:*** In this model, processors share both memory and storage disks.
- ***Shared nothing architecture:*** In this model, each processor has its own individual memory and private disk storage which may be one or more.

## 12. TERMINAL QUESTIONS

1. What do you mean by parallel database? What are the advantages and disadvantages of parallel database?
2. What are the three machine architectures upon which parallel DBMS run?
3. Discuss in detail the concept of I/O Parallelism.
4. What are the different types of partitioning techniques? Describe in detail.
5. What is Intra-query and inter-query parallelism? Explain with help of a diagram.
6. Differentiate between inter operation and intra operation parallelism.
7. What are the major design issues of parallel system?

## 13. ANSWERS

**Self-Assessment Questions**

1. True
2. (c) shared-nothing
3. True
4. granule
5. (c) Parent operations
6. Parallel-execution servers
7. True
8. Range
9. On-line transaction processing
10. True
11. Parallel query processing
12. False
13. True
14. Inter operation
15. True
16. Data

**Terminal Questions**

1. A database that has numerous memory areas and sharing only one disk drive. Refer Section 2 for more details.
2. Three machine architectures namely shared-memory, shared-disk and shared-nothing are available on which to run parallel DBMS. Refer Section 3 for more details.
3. I/O Parallelism tries to minimise the time needed to recollect relations from disk by relations partitioning on multiple disks. Refer Section 5 for more details.
4. There are three major partitioning methods: Round robin, Hash and range partitioning. Refer Section 9.5for more details
5. Intra query parallelism denotes the process of one query in parallel on multiple disc/processors. Refer Sections 6 and 7for more details
6. In intra operation parallelism, the working of each individual operation in the query is parallelised. Refer Section 8 for more details
7. The various design issues associated with parallel database are data skew, resilience to failure, parallel loading of data, etc. Refer Section 8 for more details

## 14. REFERENCES

**References:**


- Rob, P. & Coronel, C. (2004) *Database Systems: Design, Implementation, and Management,* (Sixth edition), Thomson Learning.
- Silberschatz, Korth & Sudarshan (2006) *Database System Concepts,* (Fourth edition), McGraw-Hill.
- Navathe, E. (2000) *Fundamentals of Database Systems,* (Third edition), Pearson Education Asia.


**E-references**


- http://pages.cs.wisc.edu/~anhai/courses/764-sp07-anhai/paralleldb.pdf
- http://dcx.sybase.com/1200/en/dbusage/parallelism.html
- http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.doc/admin/c0004557.htm