Databases 1

Daniel POP

Week 11

Database Performance. An Introduction

Agenda

1. Index design
    1. What indexes to create
    2. Dbms support for index performance analysis
    3. Best practices

2. Query analysis
    1. Execution plan analysis
    2. Query resource consumption

3. Table statistics

4. Database partitioning

When to create indexes

- Cost vs. benefits analysis
- Indexes inquire costs related to:
  - Disk spaces required
  - RAM
  - Fragmentation
  - Slows down INSERT/UPDATE/DELETE operations
- Benefits of an index depends on:
  - Size of table
  - Data distribution
  - Query vs. update load

When to create indexes

- Advices
  - Bigger the table, index is more valuable
  - If not often SELECT then the cost of index maintenance may be greater than the benefits
  - Data distribution – do not create indexes on Boolean or fixed set values (e.g. days of week etc)

Physical Design Advisor

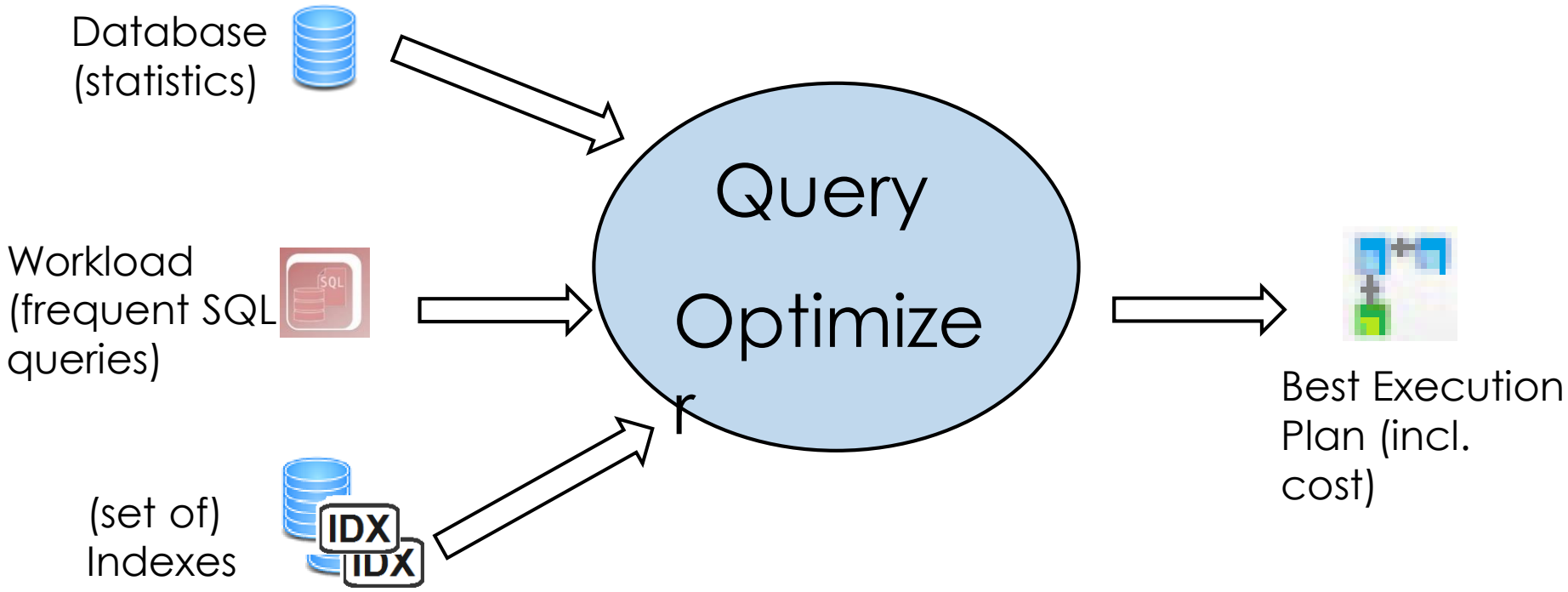
- Help users to choose what indexes to create based on (usually a sampling of) database and workload (i.e. the set of queries and updates)

- The output is a recommended set of indexes that overall optimizes performance the best

- Uses Query Optimizer to do the job.

Query Optimizer

- Component of DBMS
- It's used by Physical Design Adviser



Dynamic Management Views

- Missing indexes
  - Advice about what index MIGHT be missing
- Index usage
  - How and if the indexes are used
  - How can we tell how useful an index is?

DEMO

Microsoft Services

```
 select
      'Missing indices' as Output_Type
      , db.name as database_name
      , m.name as schema_name
      , o.name as object_name
      , [total_cost_savings]  =
              round(s.avg_total_user_cost * s.avg_user_impact * (s.user_seeks + s.user_scans),0) /100
      , s.avg_total_user_cost
      , s.avg_user_impact
      , s.user_seeks
      , s.user_scans
      , unique_compiles
      , last_user_seek
      , last_user_scan
      --, last_system_seek
      --, last_system_scan
      , d.equality_columns
      , d.inequality_columns
      , d.included_columns
from  sys.dm_db_missing_index_groups g
      inner join  sys.dm_db_missing_index_group_stats s on s.group_handle = g.index_group_handle
      inner join  sys.dm_db_missing_index_details     d on d.index_handle = g.index_handle
      inner join  sys.objects                         o on o.object_id    = d.object_id
      inner join  sys.schemas                         m on m.schema_id    = o.schema_id
      inner join  sys.databases                      db on db.database_id = d.database_id
order by total_cost_savings desc
```

Microsoft Services

```
-- Index usage
SELECT sc.name as schema_name
       , o.name as object_name
       , s.object_id
       , indexname=i.name
       , i.index_id
       , user_seeks
       , user_scans
       , user_lookups
       , user_updates
       , user_seeks + user_scans + user_lookups as total_reads
FROM sys.dm_db_index_usage_stats s
   JOIN sys.indexes  i ON i.object_id = s.object_id  AND i.index_id = s.index_id
   join sys.objects  o on o.object_id = i.object_id
   join sys.schemas sc on sc.schema_id = o.schema_id
WHERE o.type = 'U' -- user table
   and user_seeks + user_scans + user_lookups < 20
ORDER BY (user_seeks + user_scans + user_lookups) ASC
```

Microsoft

Recommendations

- Delete unused indexes
- Transform indexes (from clustered to non-clustered/columnar) to better suite the workload
- For a massive import operation, disable the impacted indexes before running the ingest process and rebuild them afterwards

Query analysis

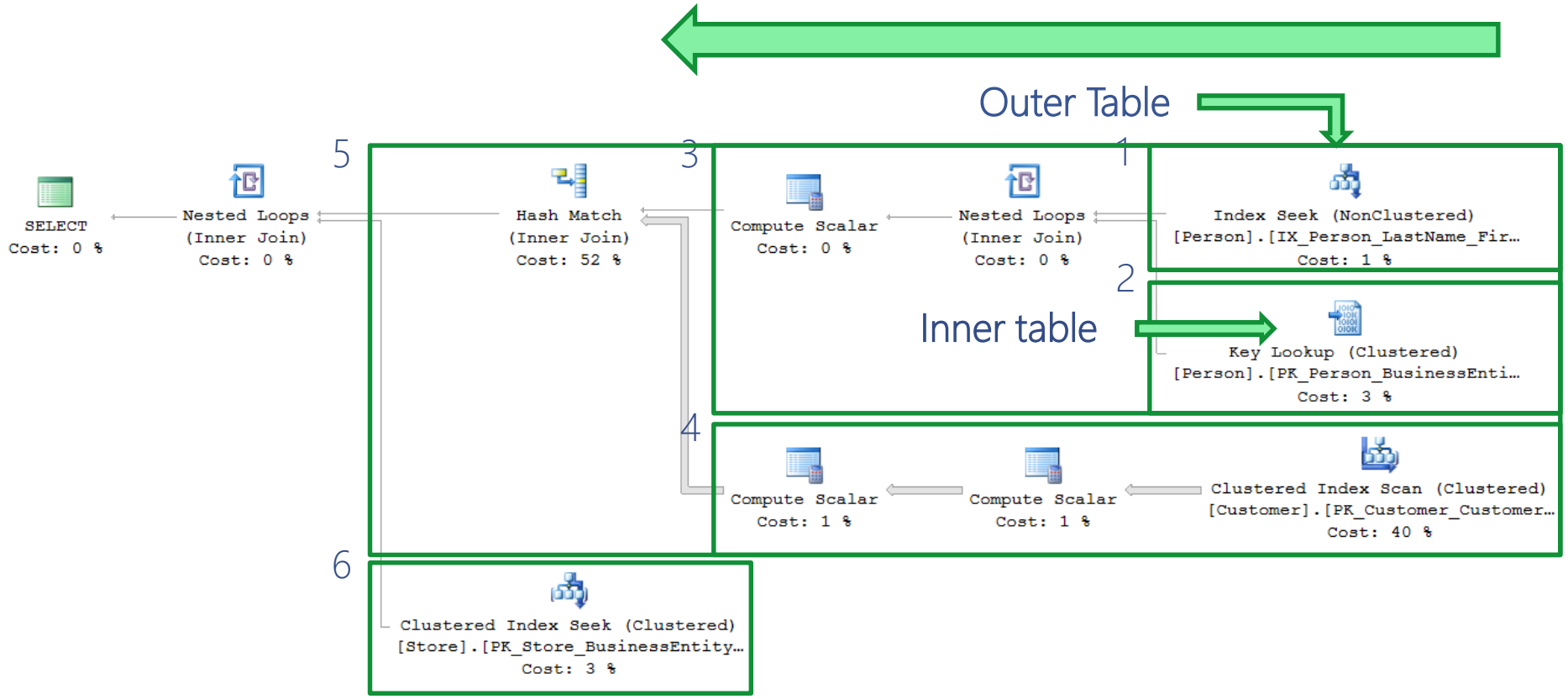
- What are the under-performant queries?

- Why is that? Where is the time spent? How can optimize them?

Execution plans

- How we execute the query (data flow)
- What methods we use to extract the data
- Runtime stats
- Statistic estimations

Graphical Showplan Flow



Resultset 1 and 2 are joined using a nested loops join, creating resultset 3
Resultset 3 and 4 are joined using a hash match join, creating resultset 5
Resultset 5 and 6 are joined using a nested loops join, creating a resultset for the Select clause

Dynamic Management Views

- Query resource consumption
  - Check for under performing queries

DEMO

Microsoft Services

```
-- Top 10 resource consuming Queries
SELECT TOP 10
    execution_count,
    statement_start_offset AS stmt_start_offset,
    total_logical_reads / execution_count AS avg_logical_reads,
    total_logical_writes / execution_count AS avg_logical_writes,
    total_physical_reads / execution_count AS avg_physical_reads,
    total_elapsed_time  / (execution_count * 1000)  AS avg_duration_ms,
    total_worker_time / (execution_count * 1000) AS avg_CPU_ms,
    total_rows / execution_count AS avg_rows_retuned,
    t.TEXT ,
    qp.query_plan
FROM
    sys.dm_exec_query_stats AS s
    CROSS APPLY sys.dm_exec_sql_text(s.sql_handle) AS t
    CROSS APPLY sys.dm_exec_query_plan(s.plan_handle) AS qp
ORDER BY
    avg_duration_ms DESC
```

Statistics

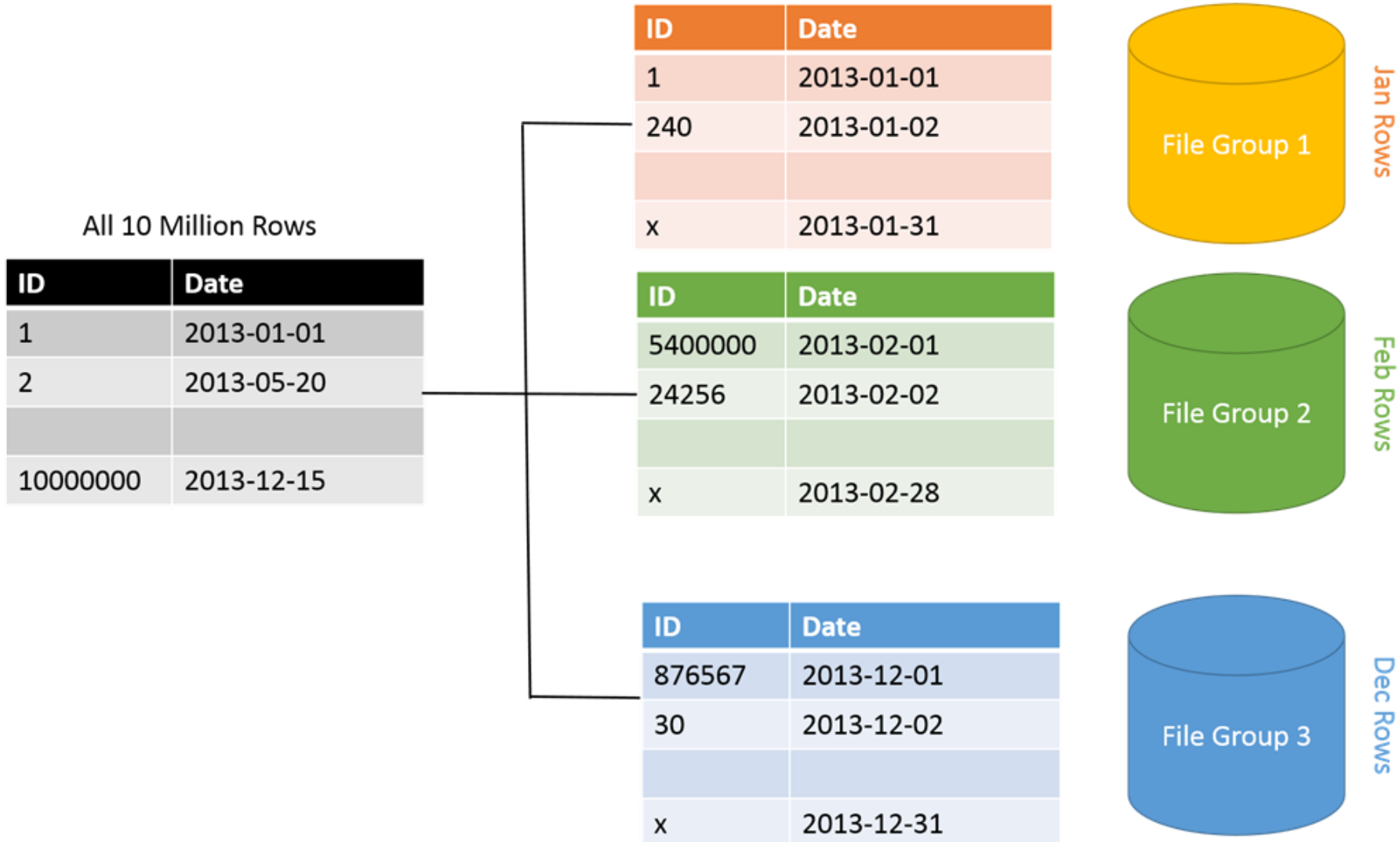
- What are statistics?
  - Distribution of values within a column
  - Density, Cardinality
- Why are statistics important?
  - Execution plan calculation
- Update statistics

DEMO

Partitioning

- Breaking a single table\index in multiple parts
- Horizontal partitioning
- Single column as partitioning key
- Per-partition management options
- Data placement on different storage
- Piecemeal backup / restore
- Dynamic scheme

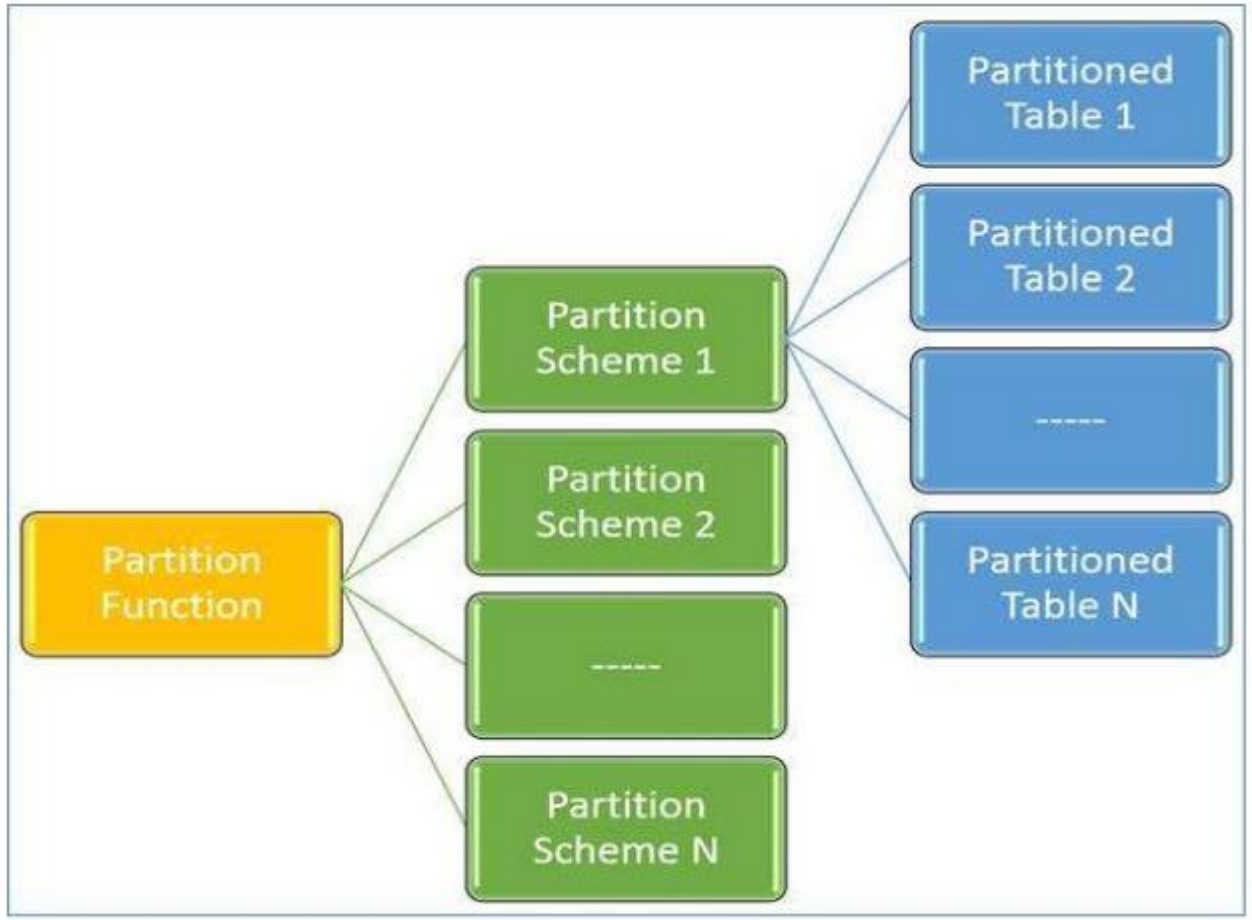
Partitioned table



Partitioning

- Each partition has its own indexes (filtered index - create a index only for some records, e.g. from one partition only)

Partition function, scheme, table



DEMO

Microsoft Services

```
-- Adds four new filegroups database
ALTER DATABASE PartiotionDemo
ADD FILEGROUP test1fg;
GO
-- etc...

-- Adds one file for each filegroup.
ALTER DATABASE PartiotionDemo
ADD FILE
(
    NAME = test1dat1,
    FILENAME = 'C:\Program Files\Microsoft SQL
Server\MSSQL14.SQL17\MSSQL\DATA\t1dat1.ndf',
    SIZE = 5MB,
    MAXSIZE = 100MB,
    FILEGROWTH = 5MB
)
TO FILEGROUP test1fg;
GO
-- etc...
```

```
-- Creates a partition function called myRangePF1 that
will partition a table into four partitions
CREATE PARTITION FUNCTION myRangePF1 (int)
    AS RANGE LEFT FOR VALUES (1, 100, 1000) ;
GO

-- Creates a partition scheme called myRangePS1 that
applies myRangePF1 to the four filegroups created above
CREATE PARTITION SCHEME myRangePS1
    AS PARTITION myRangePF1
    TO (test1fg, test2fg, test3fg, test4fg) ;
GO

-- Creates a partitioned table called PartitionTable
that uses myRangePS1 to partition col1
CREATE TABLE PartitionTable (col1 int PRIMARY KEY, col2
char(10))
    ON myRangePS1 (col1) ;
GO
```

Summary

- Indexes are the primary mechanism to improve the performance
- Implemented as hash tables or search trees
- Run a cost/benefit analysis to decide what indexes are needed
- Query planning and optimization – an important activity in relational database design
- Query Optimizer
- Table partitioning may improve the query execution time and implement a piecemeal backup strategy