Playing with a small DB and storage

Setup visualization libraries

In [1]:
```
def displaySectionCaption(caption, color='coral'):
  html_string = f'<hr><strong><p style="color:{color};font-size:16px;">{caption}</p.<
  display(HTML(html_string))
```

We study a simplified IO model for HDDs and SSDs in CMPE-138. The model will work well in practice, for our query optimzation and data layout problems.

In [2]:
```
import math
from math import ceil, log

# We'll use MBs -- for basic i to MBs
(MB, GB, TB, KB, Bytes) = (1.0, 1024.0, 1024.0*1024.0,
                           1.0/1024.0, 1.0/(1024.0*1024))

# 64 MB-Blocks (default)
PageSizeMB = 64.0*MB
size_of_types = {'int64': 8, 'int32': 4, 'double': 8, 'char': 1} # in bytes

class IOdevice:
  def __init__(self, accessTime, scanSpeed, C_w):
    self.C_r = 1.0  # Cost of reads
    self.C_w = C_w  # Cost of writes relative to reads
    self.accessTime = accessTime
    self.scanSpeed = scanSpeed

  # Read costs: Simple IOcost model using Access time + Scan speeds
  def read_pages_cost(self, numPages):
    # Assume you need to read full pages. (i.e., no partial pages)
    numPages = math.ceil(numPages)
    tsecs = numPages*self.accessTime  # time to access
    tsecs += numPages*PageSizeMB/self.scanSpeed # time to scan
    return (tsecs)

  def write_pages_cost(self, numPages):
    return self.C_w*self.read_pages_cost(numPages)

# Example IO devices in 2024
# Access and Scan speeds in [seconds, MBps], Cw cost of write vs reads.
ram1 =  (100*pow(10, -9), 100.0*1024, 1.0)
ssd1 = IOdevice(10*pow(10, -6), 5.0*1024, 1.0) # 10 microsecs, 5GBps
hdd1 = IOdevice(10*pow(10, -3), 100.0, 1.0) # 10 millissecs, 100 MBps
# machine to machine over network (modeling a network as an IO device)
m2m1 = IOdevice(10*pow(10, -6), 5.0*1024, 1.0) # 1 micro, 5 GBps

IOdevices1 = {'HDD': hdd1, 'SSD': ssd1, 'RAM': ram1}
```

In [3]:
```
"""
Basic physical table
"""
class Table:
  def __init__(self, sizeInMBs, rowSize):
    self.sizeInMBs = sizeInMBs
    self.rowSize = rowSize
    self.numRows = ceil(self.sizeInMBs/self.rowSize)

    # self.numTuples = numTuples
```

```
    self.isSorted = False
    self.isHPed = False

  # P(R) -- number of Pages for table
  def P(self):
    P = ceil(self.sizeInMBs/PageSizeMB)
    return P
  def RowSize(self):
    return self.rowSize
  def T(self):
    return self.numRows
  def SizeInMBs(self):
    return self.sizeInMBs

  # Keeping track of is table sorted, HPed, or neither (default)
  def Sort(self):
    self.isSorted = True
    self.isHPed = False
  def HP(self):
    self.isSorted = False
    self.isHPed = True
  def Reset(self):
    self.isSorted = False
    self.isHPed = False
```

Exercises:

In [25]:

```
# Spotify Songs Table [songid: int64, title: text, name: text, genre: text]
#    -- Size of row = 8 bytes (int64) + avg size of title+name+genre.
#    -- Assume avg row size = 1024 Bytes
songs_rowSize = 1024.0*Bytes
songs_numRows = 500000000.0 # 500 million songs

# """Problem 1:
# Calculate the size (MBs) of SongsTable, and num pages."""
songs_table = Table(songs_rowSize*songs_numRows, songs_rowSize)

print("The size of the table is: " + str(songs_table.SizeInMBs()) + " MB")
print("Number of pages is: " + str(songs_table.P()))
print("\n")

# """Problem 2: Read costs
# Compute the cost in seconds to read 100 pages from the SongsTable"""
for device_name, device in IOdevices1.items():
    cost = device.read_pages_cost(100)
    print("The cost for " + str(device_name) +  " is " + str(cost) + " sec")
print("\n")



# """Problem 3: Effect of caching
# Read 200 pages. 1st check RAM.
# - Cache hit of 90% in RAM.
# - For RAM cache misses (the other 10%), 75% are in SSD and 25% are in HDD."""

numPage = 200
cacheHit = 0.9
cacheMiss = 0.1
ramSSD = 75/100
ramHDD = 25/100
ram_cost = IOdevices1["RAM"].read_pages_cost(cacheHit * numPage)
ssd_cost = IOdevices1["SSD"].read_pages_cost(cacheMiss * numPage * ramSSD)
hdd_cost = IOdevices1["HDD"].read_pages_cost(cacheMiss * numPage * ramHDD)
```

```
total_cost = ram_cost + ssd_cost + hdd_cost
print("Total Cost with Caching: " + str(total_cost) + " sec")
```

```
The size of the table is: 488281.25 MB
Number of pages is: 7630


The cost for HDD is 65.0 sec
The cost for SSD is 1.251 sec
The cost for RAM is 0.06251 sec


Total Cost with Caching: 3.550168 sec
```

In [ ]: