1a)

i)

MATCH ()-[:PURCHASED]->()-[:ORDERS]->(p)

RETURN p;

ii)

MATCH (c)-[:PURCHASED]->()-[:ORDERS]->(p)<-[:SUPPLIES]-(s)

WHERE c.name = C AND p.name = P AND s.produced = S

RETURN count(\*) > 0;

iii)

MATCH (c)-[:PURCHASED]->()-[:ORDERS]->(p)-[:PART\_OF]->(cat), (p)<-[:SUPPLIES]-(s)

WHERE c.name = C

RETURN cat, s;

iv)

MATCH (c)-[:PURCHASED]->()-[:ORDERS]->(p)-[:PART\_OF]->(cat)

WHERE c.name = C

RETURN c.name, count(DISTINCT cat);

// RETURN DISTINCT (cat.name, count(cat))

v)

MATCH (c)<-[:PART\_OF|SUPPLIES|ORDERS\*1..2]-(o)

WHERE c.name = C

RETURN DISTINCT o;

Supplies direction is the other way?

1b)

i)

db.albums.find({});

ii)

db.albums.find({“pricing.list\_price”: {$gt: 1200}});

iii)

db.albums.find({“title”: /A.\*/});

iv)

db.albums.find({“details.artist”: “John Coltrane”, “shipping.weight”: {$in: [5, 8]}});

v)

db.albums.aggregate([

{$lookup: {from: “sales”, localField: “title”, foreignField: “title”, as: “sales”}}

])

2

a)

Track capacity = sector size \* track size

= 512 \* 50 = 25 600 = 25.6 KB

Surface capacity = track capacity \* number of tracks

= 25 600 \* 2000 = 51 200 000 = 51.2MB

Disk capacity = surface capacity \* number of surfaces

= 51 200 000 \* 10 = 512 MB

Disk blocks = disk capacity / block size

= 512 MB / 1024B = 500 000 blocks

Alternative:

Surface capacity = track capacity \* number of tracks

= 25 600 \* 200 = 5 120 000 = 5.12MB

Disk capacity = surface capacity \* number of surfaces

= 5 120 000 \* 10 = 51.2 MB

Disk blocks = disk capacity / block size

= 51.2 MB / 1024B = 50 000 blocks

b)

number of records per block = floor(block size / record size)

= floor(1024 / 100) = 10

number of blocks = number of records / number of records per block

= 10 000 / 10 = 1000

space wasted = number of records \* (block size - (number of records per block \* record size))

= 10 000 \* (1024 – 10 \* 100) = 10 000 \* 24B = 24 000B = 24 KB

(Unsure about the rest of the Q)

There is no wasted space if records are stored in memory as they can be stored contiguously.

50B are wasted because 1 000 000B / 75 = 13333.3 so there is 25B left over to transfer and 75B – 25B = 50B

c) (unsure)

Yes an index lookup is definitely worth it as it means we may not have to resort to reading from disk.(as much?) Random reads on disks are very expensive, so ordering a copy of this data via indexing would allow more efficient search and reads.

d)

Sequential access:

tread = tseek + ttransfer + tdelay

# sectors for data = blocks for whole file / blocks per sector = 1000/0.5 = 2000

# tracks for data = #sectors for data / track size = 2000/50 = 40  
# surfaces for data = #tracks for data / surface size = 40/200 < 1

Tseek = time to position disk head on track = seek time \* # tracks for data = 10 \* 40 = 400  
Tdelay = time to position disk head on block after rotation = delay time \* #tracks for data = 5 \* 40 = 200  
Ttransfer= time to move data from disk surface = transfer time \* #blocks = 1 \* 1000 = 1000

Tread = 400 + 200 + 1000 = 1600ms

Random access:

ASSUME track (and potentially platter, but doesn’t matter since we only consider average seek and delay times) changes at every new record.

Ttransfer remains the same

Tseek = seek time \* #records = 10 \* 10000 = 100000  
Tdelay = delay time \* #records = 5 \* 10000 = 50000

Tread = 151000

-----------------------------------------------------------------------------------------------------------

3.a. The FTL is a device driver that acts as a layer of abstraction between the OS and the flash storage device. It can tune performance of the device and help increase its lifetime.

3.b. Random access is much faster on flash than on disk, as the disk R/W arm must constantly move around the disk during random access, adding seek time + rotational delay. Sequential access will also be slightly faster on flash than on disk, since the disk will have to occasionally move the R/W head to a different track or a different sector, whereas flash doesn’t need any moving parts to read data.

Random writes are slower on flash than random reads since random writes often involve copying data, and deleting old pages in flash requires quite high voltages.

A

An erase block is bigger than a read block because, due to the high voltages involved in erasing pages, it is hard to only erase a small number of pages without also erasing the pages around them.

3.c. PCM is a good alternative

It has the advantages of a good access time and the hardware has good endurance, although the main drawback is the poor density of the data.

PCM’s key benefit over main memory is the fact that data stored in it is persistent when the device loses power.

3.d. Random writes on flash are slow for the reasons already mentioned. Also, sequential writes tend to be quite fast since they get buffered in main memory beforehand.

To improve the slow random writes, we could use the Append-Pack algorithm.