חלק ב'

1 שאלה 1

1.1 סעיף א'

SELECT DISTINCT actorId FROM playsIn WHERE character='Sheriff';

1.2 סעיף ב'

Unique (cost=616.56..616.81 rows=50 width=4) (actual time=3.294..3.389 rows=44 loops=1)
-> Sort (cost=616.56..616.69 rows=50 width=4) (actual time=3.292..3.324 rows=50 loops=1)
Sort Key: actorid Sort Method: quicksort Memory: 27kB
-> Seq Scan on playsin (cost=0.00..615.15 rows=50 width=4) (actual time=0.581..3.247 rows=50 loops=1)
Filter: (("character")::text = 'Sheriff'::text)
Rows Removed by Filter: 32602
Planning Time: 0.082 ms
Execution Time: 3.440 ms

הטבלה ממויינת ע"פ $actorId$ כמפתח מיון, ובאמצעות $quickSort$.
הזמן שנדרש עבור המיון הינו $actual\,startup\,cost = 3.292ms$
$actual\,cost\,of\,operation = 3.324ms$

$$seq\,Scan\,on\,PlaysIn \Rightarrow scans\,the\,entire\,relation\,as\,sorted\,on\,disk$$

התנאי הוא שהדמות היא $Sheriff$ כאשר הזמנים של חישוב התנאי הם:

$$actual\,startup\,cost = 0.581ms$$

$$actual\,cost\,of\,operation = 3.247ms$$

ה־$seq\,scan$ התבצע פעם אחת, והחזיר 50 שורות ולאחר הסרת הכפילויות נותרו 44 שורות.
כמו כן סה"כ שורות שלא התקיים בהם התנאי שהדמות היא $Sheriff$ הן $32,602$.
הזמן שלקח להריץ את השאילתה:

$$planning\,time = 0.082ms$$

$$execution\,time = 3.44ms$$

סה"כ: $3.522ms$.
אופן חישוב השאילתה: מעבר על כל הטבלה ,בדיקת התנאי והחזרת השורות שמתאימות לתנאי ללא כפילויות תוך שימוש במיון ע"פ $actorId$.

1.3 סעיף ג'

פקודה שתייצר אינדקס על שדה שתשפר את זמן הריצה של השאילתה:
CREATE INDEX ON playsIn(character);
נבחין כי שימוש ב־$index$ ה־$execution\,time$ יורד מ־$3.44ms$ ל־$0.443ms$.
ה־$planning\,time$ עולה מ־$0.082ms$ ל־$0.776ms$ משום שהתכנון של השאילתה כולל גישה לאינדקסים וצריך לקבוע האם השימוש בהם יהיה יעיל לפני ה־$execution\,time$.
כאשר העליה בזמן התכנון זניחה יחסית ביחס לשינוי ב־$execution\,time$.

1.4 סעיף ד'

HashAggregate (cost=122.48..122.98 rows=50 width=4) (actual time=0.274..0.316 rows=44 loops=1)
Group Key: actorid
-> Bitmap Heap Scan on playsin (cost=4.67..122.36 rows=50 width=4) (actual time=0.106..0.225 rows=50 loops=1)
Recheck Cond: (("character")::text = 'Sheriff'::text) Heap Blocks: exact=37
-> Bitmap Index Scan on playsin_character_idx1 (cost=0.00..4.66 rows=50 width=0) (actual time=0.097..0.097 rows=50 loops=1)
Index Cond: (("character")::text = 'Sheriff'::text)
Planning Time: 0.776 ms
Execution Time: 0.443 ms
(9 rows)

הזמן שלוקח להריץ את השאילתה $0.776ms + 0.443ms = 1.219ms$
אופן חישוב השאילתה
כפי שניתן לראות בשורה הכי פנימית:

$$BitmapIndexScanonplaysincharacter_idx1(cost = 0.00..4.66rows = 50width = 0)$$

מוצאים את מקום השורות המתאימות לתנאי $character =' Sheriff'$

- $cost = 0.00...4.66$ כלומר התוכנית ציפתה שהעלות תהיה 4.66, כאשר 0.00 זה ה־$startup\,time$ עבור השאילתה.
- $rows = 50$ המספר המוערך של השורות שה־$index\,scan$ יחזיר.

- $width = 0$ הוא הגודל המוערך בבתים של השורות שיוחזרו והוא 0 משום שאנו מעוניינים רק במיקום ולא בתוכן של השורות.

מכיוון שהרצנו את $EXPLAIN\ ANALYZE$, מריצים את השאילתה בפועל ומתקבל מידע נוסף על הזמנים.

$$actual\,time = 0.097...0.097,\ rows = 50,\ loops = 1$$

כלומר ה־$index\ scan$ רץ פעם אחת, החזיר 50 שורות, וה־$actual\,time$ הוא 0.

התוצאות של ה־$index\ scan$ עוברות ל־$bitmap\,heapscan$.
כעת ע"פ המיקום של השורות בטבלה $playsIn$ תחת התנאי שהדמות היא $Sheriff$ נביא אותן.

$$Bitmap\,heap\,scan\,on\,playsin(cost = 4.67...122.36\,rows = 50\,width = 4)$$

היחס בין ה־$cost$ לכל $unit = \frac{0.097ms}{4.67units} = 0.02$.

$$expected\,time = (122.36 - 4.67) \cdot 0.02 \approx 2.35ms$$

$$actual\,time\,per\,row = 0.225ms$$

כלומר, ירדנו בפקטור של בערך 10, שכן ההערכה של ה־$cost$ היא חסם עליון ואין צורך לקרוא את כל השורות.
השילוב של $bitmap\,index\,scan$ ו־$bitmap\,heap\,scan$ נחשב ליקר יותר מקריאת כל השורות בזו אחר זו מהטבלה, אך משום שיש יחסית מעט שורות שצריך לעבור עליהן, התהליך מתבצע מהר יותר.
בנוסף לכך $hash\,aggregate$ עשה שימוש ב־$hash\,table$ זמני כדי לקבץ את כל הרשומות, אין צורך במיון מקדים של ה־$data$, במקום נעשה שימוש גדול בזיכרון.

$$hash\,aggregate\left(\underbrace{cost = 122.48...122.98}_{estimated\,at\,0.5\,overall}, rows = 50, width = 4\right)$$

כאשר ה־$group\,key$ הוא $actorId$
העלות בפועל הינה $actual\,time = 0.274...0.316$

2 שאלה 2

2.1 סעיף א'

2.1.1 תת־סעיף 1

$$number\,of\,rows\,in\,block = \frac{10^3}{150} = \left\lfloor \frac{100}{15} \right\rfloor \approx 6$$

$$number\,of\,blocks\,for\,movies = \frac{10^4}{6} \approx 1667$$

צריך לבדוק עבור כל הסרטים ומה האורך שלהם
$\Leftarrow$צריך לקרוא 1667 בלוקים.

2.1.2 תת־סעיף 2

דרגת הפיצול האופטימלית של האינדקס:

pointer 8 bytes, duration 8 bytes
block size 10^3

d ילדים
$d-1$ ערכי חיפוש

$$8d + 8\,(d-1) \leq 10^3$$

$$16d - 8 \leq 10^3$$

$$16d \leq 1008$$

$$d \leq 63$$

$\Leftarrow$ דרגת הפיצול האופטימלית=63

2.1.3 תת־סעיף 3

עלות חישוב השאילתה

B+ Tree depth at most $\left\lceil \log_{\lceil \frac{d}{2} \rceil} (rows\, in\, movies) \right\rceil = \left\lceil \log_{\lceil \frac{63}{2} \rceil} \left(10^4\right) \right\rceil = 3$

Step 1: traverse tree 3

Step 2: follow leaves 1 (only a single leaf)

Total=3+1=4

2.2 סעיף ב'

2.2.1 תת־סעיף 1

$$Full\, Scan = 1667$$

2.2.2 תת־סעיף 2

דרגת פיצול:

$$\underbrace{8d}_{pointer} + \underbrace{8\,(d-1)}_{search\, key\, movie\, id} \leq \underbrace{10^3}_{block\, size} \Rightarrow 16d \leq 1008 \Rightarrow d \leq 63$$

2.2.3 תת־סעיף 3

Step 1 cost at most $\log_{32} 10^4 = 3$

matching rows $\frac{200-100}{200} \cdot 10^4 = \frac{10^6}{200} \approx 5000$

Step 2 will be in at most $\frac{matching\, rows}{\lceil \frac{d}{2} \rceil - 1} = \frac{5000}{32-1} \approx 162$

כלומר בסה"כ 3+162=165

2.3 סעיף ג'

2.3.1 תת־סעיף 1

$$Full\, Scan = 1667$$

2.3.2 תת־סעיף 2

דרגת פיצול:

$$\underbrace{8d}_{pointer} + \underbrace{8\,(d-1)}_{search\, key\, movie\, id} \leq \underbrace{10^3}_{block\, size} \Rightarrow 16d \leq 1008 \Rightarrow d \leq 63$$

2.3.3 תת־סעיף 3

עלות חישוב:

Step 1: traverse tree $\log_{\lceil \frac{d}{2} \rceil} 10^4 = 3$
Step 2: read only one relevant leaf = 1
Step 3: access matching row=1

⇐סה"כ $3 + 1 + 1 = 5$.

2.4 סעיף ד'

2.4.1 תת־סעיף 1

Full table scan = number of blocks to read = 1667

2.4.2 תת־סעיף 2

דרגת פיצול:

$$\underbrace{8d}_{pointer} + \underbrace{10\,(d-1)}_{genre} \leq \underbrace{10^3}_{block\,size} \Rightarrow 18d \leq 1010 \Rightarrow d \leq 56.11 \Rightarrow d = 56$$

2.4.3 תת־סעיף 3

Step 1: $\log_{\lceil \frac{d}{2} \rceil} 10^4 = \log_{25} 10^4 = 2.76 \approx 3$

Step 2: matching rows $\frac{1}{4} \cdot 10^4 = 2500 \Rightarrow \frac{2500}{\lceil \frac{d}{2} \rceil - 1} \approx 93$ number of leaves.

Step 3: access matching rows: $\min\,(2500, 1667)$
total: $\underbrace{3}_{depth} + \underbrace{93}_{leaves} + \underbrace{1667}_{matching\,rows} = 1763$

2.5 סעיף ה'

2.5.1 תת־סעיף 1

Full table scan = 1667

2.5.2 תת־סעיף 2

$$8d + (10 + 8)\,(d - 1) \leq 10^3$$

$$8d + 18d - 18 \leq 10^3$$

$$26d \leq 1018$$

$$d \leq 39.15 \Rightarrow d = 39$$

2.5.3 תת־סעיף 3

Step 1: $\left\lceil \log_{\lceil \frac{d}{2} \rceil} 10^4 \right\rceil = \lceil \log_{20} 10^4 \rceil = 4$

Step 2: leaves to traverse $\left\lceil \frac{matching\,rows}{\lceil \frac{d}{2} \rceil - 1} \right\rceil = \left\lceil \frac{\frac{10^4}{4}}{19} \right\rceil = \lceil \frac{2500}{19} \rceil = 132$

Step 3: No needed - all information is in index

Total $= \underbrace{4}_{depth} + \underbrace{132}_{leaves} = 136$