4.[D] Progetto fisico, contenente l’elenco degli indici che si intendono creare per le interrogazioni contenute nel carico di lavoro (specificando relazione di riferimento e chiave di ricerca), il loro tipo (ordinato/hash, clusterizzato/non clusterizzato) e la motivazione che ha portato alla loro creazione  
  
======Prima Query  
Determinare l’identificatore dei giochi che coinvolgono al più quattro squadre e  
richiedono l’uso di due dadi  
SELECT GiocoId  
FROM GIOCO  
WHERE MaxSquadre < 5 AND NrDadi = 2  
La tabella è più grande di una o poche pagine quindi possiamo tenere in considerazione gli indici.  
L’interrogazione ha una condizione di selezione di intervallo e una di uguaglianza   
Entrambe le condizioni sono fattori booleani  
I Cammini di accesso a Gioco (oltre alla scansione sequenziale) sono:  
(I\_Gioco(MaxSquadre), MaxSquadre<5)   
(I\_Gioco(NrDadi), NrDadi=2)   
(I\_Gioco(NrDadi), NrDadi=2) hash  
(I\_Gioco(MaxSquadre, NrDadi), MaxSquadre<5 AND NrDadi=2)   
(I\_Gioco(NrDadi, MaxSquadre), MaxSquadre<5 AND NrDadi=2)   
Valutiamo empiricamente quale scelta porta una miglior prestazione:  
- senza indici: 0.472 ms  
- con I\_Gioco(MaxSquadre): 0.2 ms   
- con I\_Gioco(NrDadi): 0.071 ms  
- con I\_Gioco(NrDadi): 0.091 ms hash  
- con I\_Gioco(MaxSquadre, NrDadi): 0.066 ms  
- con I\_Gioco(NrDadi, MaxSquadre): 0.039ms   
  
L’usare gli indici rispetto a non usarli ci viene confermato anche dal calcolo dei fattori di carico:  
F(MaxSquadre<5) = NrCasiFavorevoli/ NrCasiPossibili = 4/V(MaxSquadre/ Gioco) = 0.2  
F(NrDadi=2) = 1/30 = 0.03  
Entrambi sono più vicini a 0 che a 1 e la tabella è più grande di 1 o poche pagine quindi conviene usare indici.  
Inoltre il fattore di carico ci conferma anche perché otteniamo dei migliori risultati con l’indice su NrDadi. Dato che la condizione è più selettiva ha bisogno di accedere a meno tuple e quindi meno accessi a disco.  
Stessa logica per i due indici multiattributo. Il risultato intermedio con il secondo indice multiattributo è più piccolo e quindi il filtro su MaxSquadre deve agire su meno tuple.  
Per ora togliamo I\_Gioco(MaxSquadre) dato che è il più inefficiente e non viene coinvolto in altre query.  
La scelta su quale indice clusterizzare verrà fatta alla fine perché si ottiene un miglior vantaggio creandolo su un attributo coinvolto nella condizione con minore selettività. Perché quelle con minor selettività restituiscono più tuple e nel caso peggiore si rischia di fare nr tuple del risultato accessi al disco che può superare il numero di accessi a disco mediante la semplice scansione sequenziale.  
  
  
  
  
======Seconda Query  
Determinare l’identificatore delle sfide relative a un gioco A di vostra scelta (specificare direttamente  
l’identificatore nella richiesta) che, in alternativa:  
hanno avuto luogo a gennaio 2021 e durata massima superiore a 2 ore, o hanno avuto luogo a marzo 2021 e durata massima pari a 30 minuti.  
SELECT SfidaId   
FROM SFIDA  
WHERE GiocoId = 28 AND   
(  
 (Data BETWEEN '2021-01-01' AND '2021-01-31'   
 AND DurataMassima > '02:00:00')  
 OR  
 (Data BETWEEN '2021-03-01' AND '2021-03-31'  
 AND DurataMassima = '00:30:00')  
)  
La tabella è più grande di una o poche pagine quindi possiamo tenere in considerazione gli indici.  
L’interrogazione ha una condizione di selezione di uguaglianza e una condizione di selezione più complessa che ha sua volta è suddivisa in 5 condizioni di selezione di intervallo e una di uguaglianza  
Il fattore booleano è GiocoId=28 ed è il suo indice che verrà preso in considerazione come cammino di accesso dato che nelle condizioni composte il sistema preferisce scegliere un PQP che contenga un cammino di accesso basato su una condizione con indice che, se falsa, rende falsa tutta l’interrogazione.  
I cammini di accesso oltre alla scansione sequenziale sono:  
(I\_Sfida(GiocoId), GiocoId=28)  
(I\_Sfida(GiocoId), GiocoId=28) hash   
Valutiamo empiricamente se il cammino d’accesso con indice è meglio di quello sequenziale:  
senza indici: 0.7 ms  
(I\_Sfida(GiocoId), GiocoId=28): 0.021ms  
(I\_Sfida(GiocoId), GiocoId=28) hash : 0.029ms  
L’usare gli indici rispetto a non usarli ci viene confermato anche dal calcolo del fattore di carico:  
F(GiocoId=28) = 1/V(GiocoId,Sfida) si avvicina molto a 0, soprattutto per come è popolata la base di dati, e quindi la condizione è molto selettiva. Ciò spiega perché ottieniamo un miglioramento così grande con gli indici.  
Per ora manteniamo entrambi gli indici.  
La scelta, di nuovo, su quale indice clusterizzare verrà fatta alla fine.  
  
======Terza Query  
Determinare le sfide, di durata massima superiore a 2 ore, dei giochi che richiedono almeno due dadi. Restituire sia l’identificatore della sfida sia l’identificatore del gioco

SELECT Sfida.SfidaId, Gioco.GiocoId  
FROM SFIDA JOIN GIOCO ON SFIDA.GiocoId = GIOCO.GiocoId  
WHERE SFIDA.DurataMassima > '02:00:00' AND GIOCO.NrDadi > 1  
L’interrogazione ha due condizioni di selezione di intervallo  
I cammini di accesso oltre alla scansione sequenziale sono:  
(I\_Sfida(DurataMassima), DurataMassima>'02:00:00')  
(I\_Gioco(NrDadi), NrDadi>1)  
Valutiamo empiricamente se i cammini d’accesso con indice migliorano la situazione:  
- senza indici: 2.5ms di cui 1.5ms per le due scansioni sequenziali sulle due tabelle e 1ms per l’hash  
inner join  
- con indici: non cambia niente dato che il piano che usa gli indici come cammini d’accesso non viene selezionato  
Questa scelta del sistema viene giustificata dai fattori di carico:  
F(DurataMassima>’02:00:00’) = casi possibili / casi favorevoli = 132/144 = 0.91  
144 sono i diversi tempi ottenibili tra le 00:00:00 e le 23:59:59 a passi di 10 minuti quindi 00:00:00, 00:10:00, 00:20:00,.., 23:50:00  
132 sono i tempi tra le 02:00:00 e le 23:59:59 a passi di 10 minuti  
F(NrDadi>1) = casi possibili / casi favorevoli = 29/31 = 0.93  
31 perché un gioco può avere da 0 a 30 dadi ma nessuno vieta di averne di più. Questo range è considerato solo per come abbiamo popolato (in modo uniforme rispetto a tali valori) la base di dati.  
Dato che i fattori di carico sono più vicini a 1 è più conveniente la scansione sequenziale perché almeno costa tanto quanto il numero di pagine delle tabelle mentre con gli indici nel caso peggiore costa poco meno del numero di tuple delle due tabelle.  
Dato che le due condizioni di questa query sono le meno selettive nel workload e appartengono a due tabelle diverse allora sarebbe stato ragionevole usarli come chiave di ricerca per clusterizzare i relativi indici delle due tabelle. Però il sistema continua ad usare l’hash join.  
Dato che gli indici non vengono considerati allora non gli considereremo per lo schema fisico.

Elenco Indici:  
1) I\_Gioco(NrDadi, MaxSquadre) ordinato clusterizzato.  
Motivo: clusterizzato perché è l’unico indice della tabella. In particolare abbiamo scelto questo indice perché empiricamente è il più veloce ma siamo supportati anche dall’analisi col fattore di carico.  
2) I\_Sfida(GiocoId) ordinato clusterizzato  
Motivo: clusterizzato perché è l’unico indice della tabella. In particolare abbiamo scelto questo indice perché è l’unico che può essere usato come cammino d’accesso dato che la chiave di ricerca fa parte dell’unico fattore booleano.