Si consideri una base dati di una federazione sportiva che organizza tornei di scacchi:

GIOCATORE(ID, Nome, Cognome, DataNascita, Nazionalità, Elo)
TORNEO(Codice, Titolo, Città, Anno)
PARTECIPA(Torneo, Giocatore)
PARTITA(ID, CodTorneo, Data, Bianco, Nero, Risultato)

1a) Trovare nome e cognome dei giocatori di nazionalità italiana che hanno battuto giocatori con Elo maggiore o uguale di 2500 in una edizione della "Challengers Cup" precedente al 2015.

2a) Trovare nome e cognome dei giocatori che, giocando coi pezzi neri, non hanno mai battuto giocatori di cognome Polgar ad una qualunque edizione del torneo "Candidati".

1a) Le partite da considerare sono del torneo: $CP2015 = \sigma_{ANNO<2015 \wedge TITOLO:\text{"CHALLENGERS CUP"}}(TORNEO)$

Considero i giocatori italiani: $ItP = \sigma_{NAZIONALITA'=ITALIA}(GIOCATORE)$ e $P2500 = \sigma_{ELO \geq 2500}(GIOCATORI)$

$WW = \sigma_{RISULTATO::1\text{-}0}(PARTITA)$ $BW = \sigma_{RISULTATO::0\text{-}1}(PARTITA)$ $WW' = (WW \bowtie_{BIANCO::ItP.ID} ItP) \bowtie_{NERO:P2500.ID} P2500$

$BW' = (BW \bowtie_{BIANCO::P2500.ID} P2500) \bowtie_{NERO:ItP.ID} (ItP.id)$ $ItWins = (WW' \cup BW') \bowtie_{CODTORNEO::CP2015.CODICE} CP2015$

Query Finale: $Q = \pi_{NOME,COGNOME}(\sigma_{NAZIONALITA'=ITALIA}((GIOCATORE \bowtie_{GIOCATORE.ID::NERO} ItWins) \cup (GIOCATORE \bowtie_{GIOCATORE.ID::BIANCO} ItWins)))$

1b) $BPlayrPlPolgar = \sigma_{COGNOME:POLGAR}(PARTITA \bowtie_{GIOCATORE.ID::BIANCO} GIOCATORE)$ $PolLose = \sigma_{RISULTATO:0\text{-}1}(BPlayrPlPolgar)$

$LoseToPolgar = BPlayrPlPolgar - PolLose$ $LtoPinCan = \pi_{NERO}(\sigma_{TORNEO.TITOLO::CANDIDATI}(LoseToPolgar \bowtie_{CODTORNE:TORNEO.CODICE} TORNEO))$

Query finale: $Q = \pi_{NOME,COGNOME}(LtoPinCan \bowtie_{NERO::ID} GIOCATORE)$

---

2) Siano dati lo schema R=ABCDEFG e l'insieme di dipendenze funzionali
F={AC→DE, C→FB, BC→EG, B→A, A→CG, B→C, F→EG}
2a) Determinare le tre chiavi dello schema
2b) Dire se lo schema è 3NF e giustificare l'affermazione
2c) Calcolare una decomposizione ρ che ha i sottoschemi in 3NF, preserva le dipendenze e ha un join senza perdita, e descrivere il procedimento utilizzato giustificando i passaggi

NOTAZIONE { edp :: e' di troppo }

2a) DEG non appaiono mai come determinante, quindi non saranno chiavi. Inizio con:

• $ABC^+ = R \Rightarrow AB^+ = R \Rightarrow A^+ = R \Rightarrow$ A e' chiave $B^+ = R \Rightarrow$ B e' chiave.

Noto che $C \to FB \Rightarrow C \to B \Rightarrow$ C e' chiave. Chiavi = {A, B, C}

2b) Nello schema vi e' $F \to EG \Rightarrow F \to E$, F non e' superchiave ed E non e' primo, quindi viola la 3NF.

2c) Trovo una copertura minimale, $\boxed{\text{Fase 1}}$: $F = \{AC \to D, \cancel{AC \to E}^{\,C \to E}, C \to F, C \to B, \cancel{BC \to E}, \cancel{BC \to G}^{\,C \to G}, B \to A, A \to C, A \to G, B \to C, F \to E, F \to G\}$

$\boxed{\text{Fase 2}}$: $AC \to D$ rimane, D det. solo da AC. $AC \to E$ puo' essere sostituito da $C \to E$, dato che $C \to F \to E$, e così posso eliminare $BC \to E$. Sostituisco $BC \to G$ con $C \to G$ dato che $C \to F \to G$. $\boxed{\text{Fase 3}}$: Controllo le ridondanze,

• $C^+_{F/\{C \to E\}} \ni E$, $C \to E$ edp. • $C^+_{F/C \to F} \not\ni F$, rimane. $C^+_{F/\{C \to B\}} \not\ni B$, rimane. $C^+_{F/C \to G} \ni G$, $C \to G$ edp. $B^+_{F/B \to A} \not\ni A$, rimane.

$A^+_{F/A \to C} \not\ni C$, rimane. $A^+_{F/A \to G} \ni G$, $A \to G$ edp. $B^+_{F/B \to C} \ni C$, edp. le ultime due rimangono. $F: \{AC \to D, C \to F, C \to B, B \to A, A \to C, F \to E, F \to G\}$

Applico l'algoritmo. Tutti gli attributi sono compresi in F, e $\nexists X \to Y \in F \mid XY = R$, quindi ho la decomposizione:

$\rho = \{ACD, CF, BC, AB, AC, EF, FG\} \ni ACD \supset A \leftarrow$ chiave $\Rightarrow$ ha un loseless Join.

---

3) E' dato un file di 3.175.250 record. Ogni record occupa 350 byte, di cui 125 per la chiave. Un blocco contiene 2048 byte. Un puntatore a blocco occupa 4 byte. Si utilizza una organizzazione B-TREE.
3a) Calcolare l'occupazione in blocchi del file principale quando l'albero ha altezza massima.
3b) Calcolare l'occupazione in blocchi del file indice (tutti i livelli) quando l'albero ha altezza massima.
3c) Calcolare il costo di una ricerca quando l'albero ha altezza massima.

3a) record x blocco $= \left\lfloor \frac{2048}{350} \right\rfloor = 5$, blocchi x Main $= \frac{\lceil 3175250 \rceil}{5} = 635050$

3b) Quante chiavi in un blocco? $\frac{\lceil 1024 \rceil}{125+4} = 8$ chiavi → 9 puntatori. LIV 1 : $\frac{\lceil 635050 \rceil}{9} = 70562$

LIV 2 = $\frac{\lceil 70562 \rceil}{9} = 7841$ LIV 3 = $\frac{\lceil 7841 \rceil}{9} = 872$ LIV 4 = $\frac{\lceil 872 \rceil}{9} = 97$ LIV 5 $\frac{\lceil 97 \rceil}{9} = 11$ LIV 6 = 2.

File indice = 70562+7841+872+97+11+2+1 = 79386 blocchi

3c) Supponendo che la radice non sia caricata in RAM, allora sono necessari 8 accessi