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 francese che hanno battuto giocatori che si chiamano Beth Harmon in qualche torneo giocato a Boston dopo il 2010.

2a) Trovare nome e cognome dei giocatori che, giocando coi pezzi bianchi, non hanno mai battuto giocatori di Elo almeno 2300 ad una qualunque edizione del torneo "Grand Prix".

1a) $BlackWins = \pi_{NERO}(\sigma_{CITTÀ::BOSTON \wedge ANNO>2010}(\sigma_{NOME::BETH \wedge COGNOME::HARMON}(\sigma_{RISULTATO::0\text{-}1}(PARTITA \bowtie_{BIANCO::GIOCATORE.ID} GIOCATORE)) \bowtie_{CODTORNEO::CODICE} TORNEO))$

$WhiteWins = \pi_{BIANCO}(\sigma_{CITTÀ::BOSTON \wedge ANNO>2010}(\sigma_{NOME::BETH \wedge COGNOME::HARMON}(\sigma_{RISULTATO::1\text{-}0}(PARTITA \bowtie_{NERO::GIOCATORE.ID} GIOCATORE)) \bowtie_{CODTORNEO::CODICE} TORNEO))$

$FPlayer = \pi_{ID}(\sigma_{NAZIONALITÀ::FRANCIA}(GIOCATORE))$ Query finale $Q = \pi_{NOME,COGNOME}(GIOCATORE \bowtie (FPlayer \cap (BlackWins \cup WhiteWins)))$

1b) Prima trovo quelli che, hanno disputato partite da bianchi contro giocatori di elo almeno 2300 nel GrandPrix. $PTG = (PARTITA \bowtie_{CODTORNEO::CODICE} TORNEO) \bowtie_{GIOCATORE.ID::NERO} GIOCATORE$ $Partite = \sigma_{ELO \geq 2300 \wedge TITOLO::GRANDPRIX}(PTG)$

$WhiteWins = \pi_{GIOCATORE.ID}(\sigma_{RISULTATO::1\text{-}0}(Partite))$ $AllPlayer = \pi_{GIOCATORE.ID}(Partite)$ $Q' = AllPlayer - WhiteWins$

Query finale: $Q = \pi_{NOME,COGNOME}(GIOCATORE \bowtie Q')$

---

2) Siano dati lo schema R=ABCDEFG e l'insieme di dipendenze funzionali
F={ G→FA, BD→EF, CD→FA, C→B, B→DA, D→GC, C→D}
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

BCD CDG BDG BCG

2a) Noto che AEF non sono mai determinanti ⇒ non sono nella chiave. Controllo i sottoinsiemi di BCDG.
$BCD^+_F = R$, e' superchiave, controllo $BC^+_F = R$ ma $C \to B \Rightarrow C^+_F = R \Rightarrow$ C e' chiave., inoltre noto che $B \to DA$ quindi $B \to D$, e $D \to GC \Rightarrow B \to C \Rightarrow$ anche B e' chiave. Ho osservato che $D \to C$, quindi D e' chiave.

2b) ho $G \to FA \in F \Rightarrow G \to F$, ma F non e' primo e G non e' superchiave, quindi lo schema non e' in 3NF.

2c) Voglio trovare una cop. minimale, minimizzo i determinati:
$F = \{G \to F, G \to A, BD \to E, BD \to F, CD \to F, CD \to A, C \to B, B \to D, B \to A, D \to G, D \to C, C \to D\}$, vado al passo succ.:
• $BD \to E$, B e D sono chiavi, quindi ne tolgo una. • $BD \to F$ analogo. • $CD \to A$ analogo. • $CD \to F$ analogo. Ottengo:
$F = \{G \to F, G \to A, D \to E, D \to F, D \to A, C \to B, B \to D, B \to A, D \to G, D \to C, C \to D\}$, ora controllo le ridondanze:
$G^+_{F/\{G \to F\}} = \{AG\}$. $G^+_{F/\{G \to A\}} = \{FG\}$. $D^+_{F/D \to E} \not\ni E$. $D^+_{F/D \to F} \ni F$, e' di troppo. $D^+_{F/D \to A} \ni A$, e' di troppo. $B \notin C^+_{F/C \to B}$. $D \notin B^+_{F/B \to D}$. $A \in B^+_{F/B \to A}$ e' di troppo.
La copertura minimale e': $F = \{G \to F, G \to A, D \to E, C \to B, B \to D, D \to G, D \to C, C \to D\}$ applico l'algoritmo:
C'e' un attributo che non compare in $X \to Y \in F$? no $\Rightarrow \exists X \to Y \mid XY = R$? no $\Rightarrow \rho = \{FG, AG, DE, BC, BD, DG, CD\}$ contiene un elemento con una chiave, quindi ha un Join senza perdita.

---

3) E' dato un file di 2.125.800 record. Ogni record occupa 280 byte, di cui 130 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) Se l'albero ha altezza massima, i blocchi sono riempiti a meta', in un blocco entrano $\lceil \frac{1024}{280} \rceil = 4$ record.
Per il file principale servono $\lceil \frac{2125800}{4} \rceil = 531450$ blocchi.

3b) (Key∧pointer)×Block : $\lceil \frac{1024-4}{280+4} \rceil = 4 \Rightarrow 5$ puntatori a blocco. LIV 1: $\lceil \frac{531450}{5} \rceil = 106290$ LIV 2 = $\lceil \frac{106290}{5} \rceil = 21258$
LIV 3 = $\lceil \frac{21258}{5} \rceil = 4252$ LIV 4 = $\lceil \frac{4252}{5} \rceil = 851$ LIV 5 $\lceil \frac{851}{5} \rceil = 171$ LIV 6 = $\lceil \frac{171}{5} \rceil = 35$ LIV 7 = $\lceil \frac{35}{5} \rceil = 7$
LIV 8 = $\lceil \frac{7}{5} \rceil = 2$ LIV 9 = RADICE. In totale sono: 132867 blocchi

3c) Se la radice non e' in RAM, sono necessari 10 accessi.