Basi di Dati – Appello del 15/12/2022

Cognome e nome: _____ Matricola: _____ Turno: ___

Riportare sui fogli i seguenti dati: cognome, nome, matricola e turno di laboratorio.

Esame di SQL

Punteggi massimi:

- *Domande 1 e 2 svolte perfettamente: 23;*
- *Domande 1 e 3 svolte perfettamente: 25;*
- *Domande 2 e 3 svolte perfettamente: 28;*
- *Domande 1, 2 e 3 svolte perfettamente: 33.*

Lo svolgimento corretto di una sola domanda non permette il raggiungimento della sufficienza.

Le seguenti relazioni definiscono una base di dati "**A Bug's Life**" per la gestione dei bug nei progetti software.

UTENTE(Email, Nome, NotificheAbilitate)
PROGETTO(Titolo, URL, UltimaModifica)
BUG(Progetto, ApertoDa, ApertoIl, ChiusoDa*, ChiusoIl*, Descrizione, Priorità)

BUG(ApertoDa) e BUG(ChiusoDa) referenziano UTENTE(Email), BUG(Progetto) referenzia Progetto(Titolo), UTENTE(NotificheAbilitate) è un booleano.
BUG(ApertoIl), BUG(ChiusoIl) e PROGETTO(UltimaModifica) indicano Data e Ora, rispettivamente, di apertura del bug, di risoluzione del bug e di modifica del progetto. Per semplicità sono rappresentati come testo nel formato YYYYMMDDHHmm.
BUG(ChiusoDa) e BUG(ChiusoIl) possono essere NULL, cosa che significa che il bug non è ancora stato risolto (come detto in aula, si può assumere una business rule che imponga che ChiusoDa è NULL sse ChiusoIl è NULL.
BUG(Priorità) è un numero in cui la priorità massima (cioè più urgente) è il numero più alto.
Gli altri attributi sono autoesplicativi.

Domanda 1 (bassa complessità).
Ricavare, senza duplicati, i progetti che hanno bug aperti da Bill Gates prima dell'1/1/2000 e non ancora risolti. Ordinare i risultati dal progetto modificato meno recentemente a quello modificato più recentemente.

Soluzione 1.

```
SELECT DISTINCT p.Titolo
FROM progetto p JOIN bug b ON (p.Titolo=b.Progetto) JOIN utente u ON
(b.ApertoDa=u.Email)
WHERE u.Nome='Bill Gates' AND ApertoIl<'200001010000' AND ChiusoIl IS NULL
ORDER BY p.UltimaModifica;
```

Domanda 2 (media complessità).
Considerando gli utenti con indirizzo e-mail @apple.com che hanno risolto almeno due bug, visualizzare il loro nome (in ordine alfabetico) e il numero totale di progetti a cui hanno lavorato. NON usare viste né clausole WITH.
Soluzione 2.

```
-- Si noti che la query chiede il numero TOTALE di progetti e non solo quelli in
cui sono stati risolti almeno due bug
SELECT u.Nome, COUNT(DISTINCT b.Progetto)
FROM utente u JOIN bug b ON (u.Email=b.ApertoDa OR u.Email=b.ChiusoDa)
WHERE u.Email IN (
  SELECT b1.ChiusoDa
  FROM bug b1 JOIN bug b2 ON (b1.ChiusoDa=b2.ChiusoDa AND
    NOT(b1.Progetto=b2.Progetto AND b1.ApertoDa=b2.ApertoDa AND
    b1.ApertoIl=b2.ApertoIl))
  WHERE u.Email LIKE '%@apple.com')
GROUP BY u.Email
ORDER BY u.Nome;
```

Domanda 3 (alta complessità).
Tra i progetti in cui tutti i bug risolti sono stati risolti dallo stesso utente che li ha aperti, trovare il progetto che ha più bug aperti alla priorità massima (la priorità massima deve essere determinata partendo dal database). Mostrare il Titolo e l'URL di tale progetto.

Soluzione 3.

```
WITH PrioritàMassima AS (
  SELECT MAX(Priorità) AS PrioritàMax
  FROM bug),
ProgettiAutoBug AS (
  SELECT b.Progetto, COUNT(*) AS NBug
  FROM bug b JOIN PrioritàMassima p ON (b.Priorità=p.PrioritàMax)
  WHERE b.ChiusoIl IS NULL AND b.Progetto NOT IN (
    SELECT b1.Progetto
    FROM bug b1
    WHERE b1.ApertoDa<>b1.ChiusoDa))
  GROUP BY b.Progetto)
SELECT Titolo, URL
FROM ProgettiAutoBug pag JOIN Progetto p ON (pag.Progetto=p.Titolo)
WHERE NBug =
  (SELECT MAX(NBug)
   FROM ProgettiAutoBug);
```

Esame di Teoria

Domanda 1 (9 punti).
Con riferimento alla base di dati "**A Bug's Life**":

A. (4 punti) Esprimere in Algebra Relazionale l'interrogazione
Ricavare l'ultimo bug in ordine di tempo di apertura.

B. (5 punti) Esprimere, nel calcolo relazionale su tuple con dichiarazione di range, la seguente domanda:
Elencare i progetti in cui tutti i bug aperti hanno priorità 0 e sono stati tutti aperti da utenti con notifiche disabilitate.

Soluzione 1.
A. Una possibile soluzione è la seguente:

$$\pi_{B1.Progetto,B1.ApertoDa,B1.ApertoIl}(\rho_{B1\leftarrow BUG}(bug)) - \pi_{B1.Progetto,B1.ApertoDa,B1.ApertoIl} \rho_{B1\leftarrow BUG}(bug) \bowtie_{B1.ApertoIl<B2.ApertoIl} \rho_{B2\leftarrow BUG}(bug)$$

B. Una possibile soluzione è la seguente:

$$\{p.Titolo \mid p(PROGETTO) \mid \forall b(BUG)\ (b.Progetto=p.Titolo \rightarrow (b.Priorità=0 \wedge \exists u(UTENTE)\ (u.Email=b.ApertoDa \wedge u.NotificheAbilitate=false)))\}$$

Domanda 2 (7 punti).
Con riferimento alla seguente base di dati "**NerdFlix**" per gestire una piattaforma di streaming online di serie TV:

SERIE(<u>NomeSerie</u>, Produttore, Nazione, Genere)
EPISODIO(<u>NomeSerie, Stagione, Episodio</u>, TitoloEpisodio, Anno)

Vincoli di integrità referenziale:
EPISODIO(NomeSerie) referenzia SERIE(NomeSerie),
"Genere" può assumere i valori "Animazione", "Dramma", "Fantascienza" e "Commedia".

Tenendo conto dei seguenti dati quantitativi,
CARD(serie) = 1000
CARD(episodio) = 20000
VAL(Anno, episodio) = 50
VAL(Stagione,episodio) = 20

disegnare gli alberi sintattici prima e dopo l'ottimizzazione logica e calcolare il numero di tuple "mosse" prima e dopo l'ottimizzazione logica della seguente query:

$$\sigma_{\text{Stagione=19}\wedge\text{Genere='Animazione'}\wedge\text{Anno=2022}}(\text{episodio} \bowtie_{\text{EPISODIO.NomeSerie=SERIE.NomeSerie}} \text{serie})$$

Soluzione 2.
La query ottimizzata dividendo la selezione e portandola verso le foglie è:

$$(\sigma_{\text{Anno=2022}\wedge\text{Stagione=19}}(\text{episodio})) \bowtie_{\text{EPISODIO.NomeSerie= SERIE.NomeSerie}} (\sigma_{\text{Genere='Animazione'}}(\text{serie}))$$

Prima dell'ottimizzazione:
• Costo r1 = $(\text{episodio} \bowtie_{\text{EPISODIO.NomeSerie=SERIE.NomeSerie}} \text{serie})$: $10^3 \cdot 2{\cdot}10^4 = 2{\cdot}10^7$.
• Cardinalità $|r1|$= CARD(episodio) = $2{\cdot}10^4$ (equijoin attraverso la chiave esterna)
• Costo della selezione = $|r1|$
• Costo totale = $2{\cdot}10^7 + 2{\cdot}10^4 \sim 2{\cdot}10^7$.

Dopo l'ottimizzazione:

• Costo $\sigma_1 = \sigma_{Anno=2022 \wedge Stagione=19}(episodio) = CARD(episodio) = 2 \cdot 10^4$

• Costo $\sigma_2 = \sigma_{Genere='Animazione'}(serie) = 10^3$

• Tuple prodotte dalla selezione $|\sigma_1| = 1/VAL(Anno,episodio) \cdot 1/VAL(Stagione,episodio) \cdot CARD(episodio) = 1/50 \cdot 1/20 \cdot 2 \cdot 10^4 = 20$

• Tuple prodotte dalla selezione $|\sigma_2| = 1/VAL(Genere,serie) \cdot CARD(serie) = 1/4 \cdot 1000 = 250$

• Costo join $r = \sigma_1 \bowtie_{episodio.NomeSerie=serie.NomeSerie} \sigma_2 = 20 \cdot 250 = 5 \cdot 10^3$.

• Costo totale = $2 \cdot 10^4 + 10^3 + 5 \cdot 10^3 = 2{,}6 \cdot 10^4$.

Domanda 3 (9 punti).

A. Riportare la definizione di forma normale di Boyce-Codd (BCNF).

B. Esprimere la condizione necessaria e sufficiente perché una decomposizione sia senza perdita di informazione.

C. Dati R(A, B, C, D, E) e F = {A→B, C→D} dire se R è in 3FN motivando la risposta e se non lo è decomporla mostrando tutti i passaggi dicendo se il risultato è BCNF.

Soluzione 3.

A e B. Si vedano gli appunti/testo/slide.

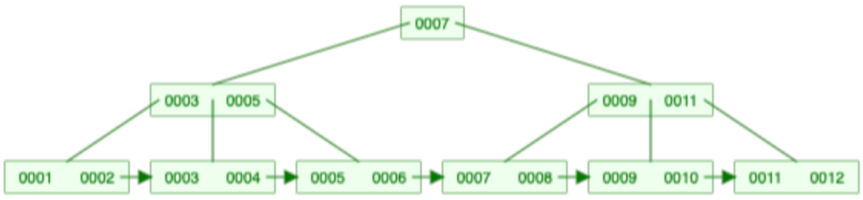
C. Per prima cosa occorre individuare la chiave della relazione R. Una possibile superchiave è ACE, che è anche l'unica chiave candidata.
Nessuna delle due d.f. dell'insieme F è in 3FN, infatti gli attributi A e C negli antecedenti da soli non sono chiave e i due attributi B e D nei conseguenti non sono parte della chiave.
Si cerca quindi l'insieme di copertura minimale di F, che in questo caso coincide con F stesso perché non ci sono né attributi estranei né d.f. ridondanti.
Si procede quindi alla normalizzazione, ottenendo le due relazioni:
R1(<u>A</u>,B)
R2(<u>C</u>,D)
A queste si deve aggiungere la relazione contenente la chiave:
R3(<u>A,C,E</u>).
Il risultato è BCNF perché tutte le d.f. (che sono sottintese) sono di tipo superchiave.

Domanda 4 (8 punti).

Rappresentare due possibili B+-tree contenenti come chiavi i numeri da 1 a 12 per i casi con m=4 e m=7. Non si richiede di simulare le singole operazioni di inserimento, ma di mostrare un possibile B+-tree con le caratteristiche indicate.

Soluzione 4.

Per m=4 ogni nodo ha al massimo m–1=3 chiavi e, tranne la radice, almeno ceil(m/2)–1=1 chiavi:



Per m=7, ogni nodo ha al massimo m–1=6 chiavi e, tranne la radice, almeno ceil(m/2)–1=3 chiavi: