1 Domande Teoria

1. Esprimere la condizione necessaria e sufficiente affinché una scomposizione sia senza perdita di informazione.

> **Solution:** Sia R(A) uno schema con dipendenze funzionali F, decomposto in $\{R1(A1), R2(A2)\}$ dove $A_1 \cup A_2 = A$
> La decomposizione di ogni istanza corretta(*) r(A) di R(A) è senza perdita di informazione se e solo se (condizione necessaria e sufficiente)
> ($A1 \cap A2$ superchiave di $A1$) $\vee$ ($A1 \cap A2$ superchiave di $A2$)

2. A proposito di gestione della concorrenza descrivere il protocollo di lock a due fasi (2PL) e quello di lock a due fasi stretto.
3. Riportare la definizione di protocollo 2PL (two-phase lock) stretto
4. Dire cos'è e quali problemi risolve il protocollo 2PL (due fasi) stretto.

> **Solution:** Il $2PL$ è un protocollo in cui le transazioni acquisiscono i **lock** nel tempo, ma, una volta **rilasciato almeno un lock**, non se ne potranno più acquisire. Si chiama **Two-Phase** perché le due fasi sono quella di **acquisizione** e **rilascio**.
> Il $2PL$ stretto è un **protocollo a due fasi modificato**. Le differenze stanno nella seconda fase, quando si inizia il rilascio dei **lock**:
>
> - T_i acquisice **lock** finché può (sia LS che LX)
> - Quando inizia il rilascio può solo rilasciare i LS
> - I LX verranno rilasciati solo al termine (**commit** o **rollback**)
>
> Questa soluzione, più rigida del $2PL$, risolve il problema dei **rollback a cascata** in quanto le altre transazioni non hanno avuto accesso agli oggetti **modificati** da T_i

5. Dire in cosa consiste la tecnica del dump/restore (ripresa a freddo) e quando si rende necessaria.
6. Esporre la differenza tra indici densi ed indici sparsi

> **Solution:** Gli **Indici Densi** sono indici con tante chiavi di ricerca quanti sono i valori distinti dell'attributo su cui sono costruiti.
> Gli **Indici Sparsi** sono indici in cui nelle foglie non ci sono più tutti i valori distinti della chiave di ricerca ma **solo un valore** (il **massimo**).

7. Riportare la definizioe di chiusura di un insieme di attributi

> **Solution:** La chiusura di un insieme di dipendenze funzionali F è un insieme di dipendenze funzionali F^+ tali che ogni dipendenza funzionale f^+ dell'insieme F^+ sia derivabile da F
> (l'insieme F^+ è finito)

8. Spiegare la differenza tra ottimizzazione logica ed ottimizzazione fisica di un'interrogazione

Solution: L'ottimizzatore logico lavora sull'albero di parsificazione
L'ottimizzatore fisico considera l'albero prodotto dall'ottimizzazione logica e scegliere gli algoritmi da abbinare ai nodi operativi

9. Dare la definizione di insieme di copertura minimale

Solution: Un insieme F' di dipendenze funzionali è un insieme di copertura minimale rispetto ad F quando:

1. F' è equivalente ad F
2. ogni $X \to Y \in F'$ è in forma canonica, cioè Y è un attributo
3. ogni $X \to Y \in F'$ è priva di attributi estranei
4. ogni $X \to Y \in F'$ non è ridondante

10. Riportare la definizione di BCNF

Solution: BCNF è una **forma normale** per una relazione r in cui ogni dipendenza funzionale (non banale, $A \to A$) $X \to A$ definita su di essa, X contiene una chiave K di r, cioè X è superchiave per r.

1. $Y \subseteq X (X \to Y$ è **riflessiva**)
2. X è **superchiave**

11. Elencare e spiegare brevemente le proprietà ACID delle transazioni

Solution:

Atomicità (Gestore del Ripristino)
il processo deve essere suddivisibile in un numero finito di unità indivisibili, chiamate transazioni. L'esecuzione di una transazione perciò deve essere per definizione o totale o nulla, e non sono ammesse esecuzioni parziali; un processo, anche parziale, invece, in quanto insieme di transazioni può non essere elementare.

Consistenza (Gestore delle Trasazioni e Serializzatore)
il database rispetta i vincoli di integrità, sia a inizio che a fine transazione. Non devono verificarsi contraddizioni (incoerenza dei dati) tra i dati archiviati nel DB;

Isolamento (Serializzatore)
ogni transazione deve essere eseguita in modo isolato e indipendente dalle altre transazioni, l'eventuale fallimento di una transazione non deve interferire con le altre transazioni in esecuzione;

Durabilità (Gestore del Buffer)
detta anche persistenza, si riferisce al fatto che una volta che una transazione abbia richiesto un commit work, i cambiamenti apportati non dovranno essere più persi. Per evitare che nel lasso di tempo fra il momento in cui la base di dati si impegna a scrivere le modifiche e quello in cui li scrive effettivamente si verifichino perdite di dati dovuti a malfunzionamenti, vengono tenuti dei registri di log dove sono annotate tutte le operazioni sul DB.

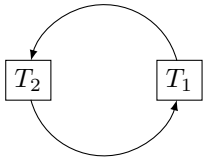
12. Definire la nozione di azioni in conflitto in una storia S

Solution: Le azioni di una storia S sono in conflitto quando:

1. appartengono a diverse transazioni,
2. operano sullo stesso valore e
3. **almeno una** delle due è una operazione di **write**

13. Descrivere il problema del deadlock avvalendosi del grafo di attesa

Solution:

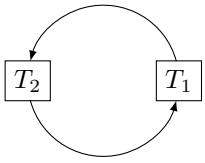


In questa situazione T_1 è in attesa di un rilascio da parte di T_2 e T_2 è in attesa di un rilascio da parte di T_1.
Il grafo di attesa presenta un **ciclo**, e quando ce n'è uno ci troviamo in una situazione di deadlock.

14. Presentare una tecnica di **superamento** del deadlock

Solution:

- Analisi del grafo di attesa (**NON Grafo dei conflitti**)



- **Timeout**
  Non viene costruito nessun grafo particolare, ma viene valutato da quanto tempo una transazione è in **wait**.
  Se è in wait da molto tempo (è in **timeout**), il DBMS la abortisce **SUPPONDENDO** che la transazione sia in **deadlock**. In questo modo si rompe il ciclo delle attese.

- **Timestamp**
  Non richiede l'analisi di grafi di attesa, ma **PREVIENE** i deadlock. Questa tecnica è basata sui **timestamp** (tempo di arrivo di una transazione nel sistema).
  Complessivamente in questo sistema non può mai formarsi il ciclo del grafo delle attese, perché gli archi di attesa possono andare solo da transazioni giovani verso transazioni vecchie.
  Se una transazione più giovane arriva e acquisice il lock su X e, successivamente, quella vecchia ha bisogno di X, allora il DBMS fa il **rollback** della giovane per far eseguire la vecchia.

15. Presentare una tecnica di **prevenzione** del deadlock

Solution:

Timestamp
Non richiede l'analisi di grafi di attesa, ma **PREVIENE** i deadlock. Questa tecnica è basata sui **timestamp** (tempo di arrivo di una transazione nel sistema).
Complessivamente in questo sistema non può mai formarsi il ciclo del grafo delle attese, perché gli archi di attesa possono andare solo da transazioni giovani verso transazioni vecchie.
Se una transazione più giovane arriva e acquisice il lock su X e, successivamente, quella vecchia ha bisogno di X, allora il DBMS fa il **rollback** della giovane per far eseguire la vecchia.

16. Definire il concetto di dipendenza funzionale

Solution: Prendiamo come esempio una relazione cittadino in cui dal Codice Fiscale riesco a dedurre tutti gli altri attributi (nome, cognome, data di nascita, ecc.)
CF $\rightarrow$ nome, cognome, data_nascita.
Ogni volta che vengono prese in esame tuple distinte di una relazione e le tuple coincidono sul valore del Codice Fiscale, impongo al DBMS che anche tutti gli attributi caratterizzanti il cittadino coincidano.

17. Enunciare il criterio di serializzabilità

Solution: Se partiamo da storie corrette (le diverse esecuzioni seriali) e abbiamo una storia interfogliata, possiamo concludere che la storia interfogliata è corretta se è **view-equivalente** ad una qualsiasi storia seriale.
Il criterio di serializzabilità dice quindi che una storia **interfogliata** S è corretta se è **view-equivalente** ad una storia seriale qualsiasi delle transazioni coinvolte da S
N.B.: date n transazioni esistono $n!$ storie seriali

18. Differenze tra $B-tree$ e B^+-tree e vantaggi dei B^+-tree rispetto ai $B-tree$ nella gestione degli indici

Solution: L'unica differenza sta nelle foglie che nel B^+-tree hanno una chiave in più (che è la chiave **separatrice** che trovo risalendo l'albero). Questo permette di collegare tra loro le foglie (quindi gli indici contenute in esse) e ciò permette di scandire sequenzialmente le foglie. Questo velocizza di molto le **ricerche di range di valori**.

19. Mostrare un esempio semplice (una relazione R ed un insieme di dipendenze funzionali F) per cui l'algoritmo di normalizzazione in BCNF non può essere in grado di mantenere la località delle dipendenze.
20. Che cosa si intende per **tupla spuria**
21. Chiarire il concetto di decomposizione con join con o senza perdita di informazione
22. Enunciare la condizione necessaria e sufficiente di decomponibilità senza perdita di informazione di una relazione in due sottorelazioni
23. Enunciare il criterio di view-serializzabilità di una storia
24. Definire il grafo dei conflitti ed enunciare la condizione sufficiente di serializzabilità
25. Quando una storia è *conflict-serializable*?

> **Solution:** Per verificare che una storia sia *conflict-serializable* bisogna costruire il grafo dei conflitti e verificare che **sia aciclico**. In questo caso la storia è anche *view-serializable*.

26. Indicare almeno **due casi** in cui gli indici secondari si dimostrano inefficienti
27. Indicare almeno tre casi in cui è preferibile evitare la definizione di indici secondari

> **Solution:**
>
> 1. Evitare gli indici su tabelle di poche pagine
> 2. Evitare indici su attributi volatili (ad esempio, Saldo del conto corrente)
> 3. Evitare indici su chiavi poco selettive
> 4. Evitare indici su chiavi con valori sbilanciati
> 5. Limitare il numero di indici
> 6. Definire indici su chiavi relazionali ed esterne
> 7. Gli indici velocizzano le scansioni ordinate
> 8. Usare l'indice hash solo per interrogazioni puntuali

28. Disegnare un $B^+ - tree$ con $m = 4 \ldots$

> **Solution:**
>
> - tutti i nodi **due o più chiavi**, eccetto la **radice**
> - ogni chiave k dei nodi **intermedi** deve contenere il valore più a destra del sottoramo sinistro.
> - se un nodo ha j chiavi deve avere $j + 1$ figli

Dubbi

1. perdita della località delle dipendenze funzionali?
2. Conflict-serializable
3. view-serializable
4. EXISTS, NOT EXISTS
5. IN, NOT IN
6. EXCEPT