1. [Esame](#Cheats)
2. [Definizioni](#Definizoni)
3. [Algebra Relazionale](#AlgebraRelazionale)

**Esame**

1. [(Esercizio 1) Traduzione modello E/R - Relazionale](#ModelloER)
2. (Esercizio 2) Algebra Relazionale
3. (Esercizio 2) Definisci un meccanismo
4. (Esercizio 2) Algebra Relazionale
5. [(Esercizio 3) Formulario B+ Tree](#BPlusTree)
6. [(Esercizio 3) Esercizio Transazioni](#Transazioni)
7. [(Esercizio 4) NoSQL appunti e teoria](#NoSQL)

**Modello E/R – Logico Relazionale**

Quando si passa dal modello concettuale al modello logico, le chiavi (e anche eventuali attributi) delle relazioni **1:1, 1:n** e **n:n** vengono accorpate in modo diverso a seconda del tipo di relazione.

* **Relazioni 1:1**

In una relazione 1:1 è possibile accorpare le chiavi primarie delle due tabelle. La chiave primaria di una delle due tabelle andrà nell’altra (a scelta, preferibilmente quella che **NON** ha una cardinalità minima di 0) insieme agli eventuali attributi

* **Relazioni 1:n**

In una relazione 1:n la chiave primaria tella tabella **n** andrà nella tabella **1** insieme agli eventuali attributi

* **Relazioni n:n**

Nelle relazioni n:n viene sempre creata una nuova tabella a parte appositamente per accoppiare la chiave primaria della prima e della seconda e gli eventuali attributi

1. Definire un meccanismo per far si…
2. Scrivere un’espressione in algebra relazionale
3. Scrivere le seguenti query

**Formule Per i B+ Tree**

**FORUMLE:**

**Esempi**

Lk = 50 Byte

Nt = 600.000

Lt = 400 Byte

Lp = 2048 Byte

Puntatori = 4 Byte

m = ogni nodo non foglia dell’albero ha al più m figli

V = numero di valori assunti dalla chiave

Lk = lunghezza della chiave

1. Calcolare la Profondità minima ()

= =

= 4

1. Costo di ricerca di una tupla

**Transazioni**

1. [Determinare se lo schedule è c-serializzabile](#CSerializzabile)
2. [Indicare le anomalie associate allo schedule](#AnomalieSchedule)
3. [Determinare l’ordine di esecuzione delle operazioni con il 2PL stretto](#Applica2PLStretto)

**Determinare se C-Serializzabile**

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, diagramma, linea, Carattere

Descrizione generata automaticamente

**Anomalie associate allo schedule:**

* Dirty Read (Lettura sporca): Si verifica quando una transazione legge dati modificati da un'altra transazione che successivamente viene annullata. In altre parole, una transazione legge dati non ancora confermati o permanenti.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

* Non-Repeatable Read (Lettura non ripetibile): Si verifica quando una transazione legge un dato più volte durante la sua esecuzione e ottiene valori diversi nelle diverse letture.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

* **Phantom Read** (Lettura fantasma): Si verifica quando una transazione esegue una Read, ma durante l'esecuzione della transazione un'altra transazione effettua delle Write che vanno a modificare il valore letto dalla prima transazione

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

* **Lost Update** (Perdita di aggiornamento): Si verifica quando due o più transazioni leggono contemporaneamente lo stesso dato e successivamente lo aggiornano. Solo una delle transazioni è in grado di confermare le proprie modifiche, mentre le altre perdono l'aggiornamento effettuato.

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente

1. **Determinare l’ordine di esecuzione delle operazioni con il 2PL stretto**

Specificare i **r\_lock**, **w\_lock** e eventuali wait.

Ricorda che nel 2PL stretto gli unlock vanno rigorosamente dopo il COMMIT

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente (NO = wait)

**NoSQL**

1. [Vantaggi e svantaggi](#VantaggiESvantaggi)
2. [Embedding](#Embedding)
3. [Referencing](#Referencing)

Gli schemi di database NoSQL (Not Only SQL) sono progettati per gestire volumi di dati elevati, modelli di dati flessibili e requisiti di scalabilità orizzontale.

**Vantaggi dei database NoSQL:**

* **Scalabilità orizzontale**: I database NoSQL sono progettati per scalare orizzontalmente, il che significa che possono gestire grandi volumi di dati distribuendo il carico su più server.
* **Flessibilità del modello dati**: I database NoSQL consentono di memorizzare dati in diversi formati come documenti, colonne, grafi o chiavi-valore.
* **Prestazioni elevate**: Poiché i database NoSQL sono progettati per operazioni di lettura/scrittura su larga scala, possono fornire prestazioni elevate per determinati carichi di lavoro.

**Svantaggi dei database NoSQL:**

* **Mancanza di supporto per le transazioni complesse**: A differenza dei database relazionali, molti database NoSQL non supportano transazioni ACID complesse che coinvolgono più operazioni.
* **Limitata flessibilità nelle interrogazioni**: Poiché i database NoSQL sono progettati per modelli di dati specifici, potrebbero non essere adatti a query complesse che richiedono operazioni di join o aggregazioni complesse tra più entità.
* **Maturità dello strumento**: Molti database NoSQL sono relativamente nuovi rispetto ai database relazionali, quindi potrebbero non avere la stessa maturità, stabilità o ampiezza di strumenti e librerie disponibili.

**Embedding**

L'**embedding** è un meccanismo in cui un documento o un oggetto viene incorporato all'interno di un altro documento come parte dei suoi dati. In pratica, i dati correlati vengono inseriti direttamente all'interno del documento principale. Questo consente di avere tutte le informazioni necessarie in un'unica struttura dati e semplifica l'accesso ai dati correlati.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamenteAd esempio, si potrebbero incorporare i dati di un prodotto, nel documento relativo all’ordine stesso

**Referencing**

Il **referencing** è invece un meccanismo in cui due documenti o oggetti sono collegati tra loro attraverso un identificatore unico. Invece di incorporare i dati correlati direttamente nel documento principale, si fa riferimento ad altri documenti tramite un identificatore o una chiave esterna. Questo permette di mantenere i dati correlati separatamente e di referenziarli quando necessario.

Ad esempio, in un database NoSQL basato su chiave-valore, si potrebbe avere un documento principale che rappresenta un ordine e una chiave esterna che fa riferimento a un documento separato che contiene i dettagli del cliente.

**Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

**Definizioni**

1. [Cos’è il modello E/R?](#ModelloERSpiegazione)
2. [Cos’è il modello Relazionale?](#ModelloRelazionale)
3. [Cos’è la progettazione Concettuale?](#ProgettazioneConcettuale)
4. [Cos’è la progettazione Logica?](#ProgettazioneLogica)
5. [Cos’è la progettazione Fisica?](#ProgettazioneFisica)
6. [Cos’è un B+ Tree?](#BPlusTreeSpiegazione)
7. [Cosa sono le Transazioni?](#TransazioniSpiegazione)

**Cos’è il modello E/R?**

In sintesi, il modello E/R è uno strumento di progettazione concettuale utilizzato nella progettazione dei database che permette di rappresentare in modo chiaro e intuitivo i concetti e le relazioni all'interno di un'applicazione, fornendo una base solida per la creazione di database efficaci e coerenti.

* Entità: Rappresenta un oggetto o un concetto distintivo all'interno del dominio di un'applicazione che può avere attributi che descrivono le sue caratteristiche.
* Relazione: Rappresenta l'associazione o la connessione tra due o più entità nel dominio.

**Cos’è il modello Relazionale?**

Il modello relazionale è un modello di dati che organizza le informazioni in tabelle strutturate chiamate relazioni.

Nel modello relazionale, i dati sono rappresentati come insiemi di tabelle o relazioni. Ogni tabella ha un nome univoco e una struttura definita da un insieme di colonne (attributi) e righe (tuple) che rappresentano le istanze dei dati. Ogni attributo ha un nome univoco e un tipo di dato associato che definisce il tipo di valore che può essere memorizzato in quella colonna.

Le relazioni nel modello relazionale sono stabilite attraverso l'utilizzo delle chiavi primarie (Primary Key - PK) e delle chiavi esterne (Foreign Key - FK). Una chiave primaria è un attributo (o una combinazione di attributi) che identifica univocamente ogni riga nella tabella. Una chiave esterna è un attributo che fa riferimento alla chiave primaria di un'altra tabella, stabilendo una relazione tra le tabelle.

**Cos’è la progettazione Concettuale?**

La progettazione concettuale è la prima fase della progettazione di un database ed è focalizzata sulla creazione di un modello concettuale che rappresenti i requisiti e la struttura logica dei dati in un modo indipendente dal sistema di gestione dei database (DBMS) specifico.

Durante la progettazione concettuale, vengono identificate le entità principali coinvolte nel dominio di un'applicazione, le loro relazioni e gli attributi che le descrivono. L'obiettivo principale è catturare il significato e le interconnessioni dei dati, piuttosto che la loro implementazione tecnica.

La progettazione concettuale coinvolge diverse attività:

1. Raccolta dei requisiti: Vengono identificati i concetti chiave, le regole di business e le relazioni tra le informazioni.
2. Identificazione delle entità: Vengono identificate le entità principali nel dominio dell'applicazione, come persone, oggetti, eventi o concetti rilevanti.
3. Definizione delle relazioni: Vengono identificate le connessioni o le associazioni tra le entità. Si determina se le relazioni sono di tipo 1:1, 1:n o n:n e si stabiliscono le cardinalità delle relazioni.
4. Attributi delle entità e delle relazioni: Vengono identificati gli attributi che descrivono le caratteristiche delle entità e delle relazioni.
5. Normalizzazione: Si applicano le regole di normalizzazione per garantire la ridondanza dei dati e l'eliminazione delle anomalie. La normalizzazione aiuta a strutturare correttamente il modello concettuale e a migliorare l'efficienza e l'integrità dei dati.

**Cos’è la progettazione Logica?**

La progettazione logica è la seconda fase della progettazione di un database ed è incentrata sulla traduzione del modello concettuale.

Durante la progettazione logica, si prende il modello concettuale e si converte in uno schema logico costituito da tabelle, colonne, chiavi primarie, chiavi esterne e altre restrizioni di integrità dei dati.

1. Traduzione delle entità: Ogni entità nel modello concettuale viene mappata in una tabella del database. Gli attributi dell'entità diventano le colonne della tabella.
2. Definizione delle chiavi primarie: Per ogni tabella, viene identificata una chiave primaria che identifica univocamente ogni riga nella tabella. La chiave primaria può essere costituita da uno o più attributi.
3. Definizione delle relazioni: Le relazioni tra le entità vengono trasformate in collegamenti tra le tabelle attraverso l'utilizzo delle chiavi primarie e delle chiavi esterne.
4. Normalizzazione: Si applicano le regole di normalizzazione per eliminare la ridondanza dei dati e garantire l'integrità dei dati.
5. Definizione delle restrizioni di integrità: Vengono imposte restrizioni per garantire l'integrità e la coerenza dei dati, come restrizioni di chiavi esterne, restrizioni di unicità etc.

**Cos’è la progettazione Fisica?**

La progettazione fisica è la fase finale della progettazione di un database ed è incentrata sull'implementazione concreta dello schema di database definito nella progettazione logica su un sistema di gestione dei database (DBMS) specifico.

La progettazione fisica si occupa di tradurre lo schema logico in strutture di archiviazione fisiche, definendo gli indici, le tabelle, le viste, le partizioni e altri aspetti tecnici del database.

Durante la progettazione fisica, vengono prese decisioni specifiche sulle caratteristiche di implementazione, tenendo conto delle prestazioni, dell'efficienza e delle considerazioni di utilizzo del sistema.

1. Definizione delle tabelle fisiche: Vengono create le tabelle fisiche nel database, stabilendo i tipi di dati appropriati per le colonne e la dimensione dei campi. Si definiscono le proprietà di archiviazione, come la dimensione dei blocchi, i file di dati e i file di log.
2. Creazione degli indici: Si identificano le colonne che saranno spesso utilizzate per la ricerca e si creano gli indici appropriati per migliorare le prestazioni delle query. Gli indici accelerano la ricerca e il recupero dei dati, creando strutture di dati aggiuntive per consentire un accesso rapido.
3. Ottimizzazione delle query: Si valutano le query più frequenti e si applicano tecniche di ottimizzazione per migliorare le prestazioni. Ciò può coinvolgere la ridefinizione delle query, la creazione di viste materializzate o l'aggiunta di indici specifici.
4. Partizionamento dei dati: Se il volume dei dati è significativo, si possono utilizzare tecniche di partizionamento per distribuire i dati su più dispositivi di archiviazione o per suddividere i dati in base a criteri predefiniti, migliorando così l'efficienza delle operazioni di accesso e query.
5. Sicurezza e autorizzazioni: Si definiscono i livelli di sicurezza e si assegnano le autorizzazioni di accesso agli utenti e ai ruoli. Questo assicura che solo gli utenti autorizzati possano accedere ai dati e protegge l'integrità e la riservatezza delle informazioni.

La progettazione fisica comprende anche aspetti come la replica dei dati, la ridondanza, la gestione della concorrenza e la pianificazione delle transazioni per garantire l'affidabilità, la disponibilità e la performance del database.

**Cos’è un B+ Tree?**

Un B+ tree (albero B+) è una struttura dati ad albero bilanciato utilizzata principalmente per l'organizzazione e la gestione efficiente di grandi quantità di dati in memoria o su disco

Un B+ tree è composto da nodi che possono contenere un numero variabile di chiavi ordinate. Le chiavi sono utilizzate per organizzare e recuperare i dati. Ogni chiave nel B+ tree è unica e associata a un valore (record) corrispondente.

T le chiavi sono memorizzate nei nodi foglia. I nodi interni del B+ tree contengono solo puntatori alle chiavi nei nodi foglia. Ciò rende i nodi interni più piccoli e permette di memorizzare più chiavi nel buffer di memoria.

**Cosa sono le Transazioni?**

Le transazioni in basi di dati sono una sequenza di operazioni che devono essere eseguite come un'unità atomica e indivisibile. Una transazione rappresenta un'azione o una serie di azioni che modifica lo stato di una base di dati da uno stato coerente a un altro.

Le transazioni devono soddisfare le proprietà ACID, che sono:

* **Atomicità**: Una transazione viene considerata come un'unità atomica indivisibile, il che significa che tutte le sue operazioni devono essere eseguite completamente o nessuna di esse deve essere eseguita affinché la transazione venga considerata completata con successo.
* **Coerenza**: Una transazione deve portare il sistema da uno stato coerente a un altro stato coerente. In altre parole, le transazioni devono rispettare i vincoli di integrità dei dati definiti nel database.
* **Isolamento**: Le transazioni devono essere eseguite in modo isolato, il che significa che le modifiche apportate da una transazione non devono essere visibili ad altre transazioni fino a quando la transazione stessa non viene completata con successo.
* **Durabilità**: Una volta che una transazione viene completata con successo, le sue modifiche devono essere permanenti e persistenti nel sistema, anche in caso di guasto del sistema o riavvio successivo.

**Algebra Relazionale**

1. [Operatori di Base](#OperatoriDiBase)
   1. [Unione](#Unione)
   2. [Differenza](#Differenza)
   3. [Intersezione](#Intersezione)
   4. [Prodotto Cartesiano](#ProdottoCartesiano)
2. [Altri operatori](#AltriOperatori)
   1. [Selezione](#Selezione)
   2. [Proiezione](#Proiezione)
   3. [Ridenominazione](#Ridenominazione)
   4. [Join Naturale](#JoinNaturale)
   5. [Left Join](#LeftJoin)
   6. [Right Join](#RightJoin)
   7. [Theta Join](#ThetaJoin)
   8. [Equi Join](#EquiJoin)
3. [Esempi di algebra relazionale](#EsempiDiAlgebraRelazionale)

*Se due insiemi e hanno lo stesso numero e nomi di attributi si dice che A e B hanno “lo stesso schema”*

**Operatori di base**

**Unione**:

*Solo se hanno lo stesso schema. Restituisce tutti i record presenti in uno dei due insiemi o in entrambi*

**Differenza:**

*Solo se hanno lo stesso schema. Restituisce tutti i record presenti nel primo insieme, ma non nel secondo*

**Intersezione:**

*Solo se hanno lo stesso schema. Restituisce tutti i record che sono presenti in entrambi gli insiemi*

**Prodotto cartesiano:**

*Il prodotto cartesiano tra A e B è l’insieme di tutte le coppie ordinate in cui* ***a*** *appartiene ad* ***A*** *e* ***b*** *appartiene a* ***B***

**Altri Operatori**

Selezione:

*(Selezione orizzontale)*

Prende una condizione; è unario prende le tuple della relazione che rispettano la condizione, filtra le tuple

Proiezione:

*(Selezione verticale)*

Filtra gli attributi risultato con stesse tuple di A ma meno attributi (solo e )

Ridenominazione:

Rinomina un attributo (colonna) con il nome specificato, utile per creare copie della stessa relazione

**Join Naturale:** ⨝

Mette insieme le tuple di 2 relazioni in base agli attributi con lo stesso nome

*(Se non ci son attributi in comune è uguale al prodotto cartesiano)*

**Left Join:**

Restituisce tutte le righe dalla tabella di sinistra (**left**) e solo le righe corrispondenti dalla tabella di destra (**right**) in base alla condizione di join specificata. Se non c'è una corrispondenza, le colonne della tabella di destra saranno riempite con valori nulli.

**Right Join:**

È l'opposto del left join e restituisce tutte le righe dalla tabella di destra (right) e solo le righe corrispondenti dalla tabella di sinistra (left) in base alla condizione di join specificata. Se non c'è una corrispondenza, le colonne della tabella di sinistra saranno riempite con valori nulli.

**Theta Join:**

Il theta join è un join che utilizza una condizione di join arbitraria, specificata dall'utente, come ad esempio una condizione di confronto tra le colonne delle due tabelle coinvolte. Il theta join può essere eseguito utilizzando gli operatori di confronto come '=', '>', '<', '<=', '>=' e '<>'.

**Equi Join:** l'equi join è un tipo di join che utilizza la condizione di uguaglianza ('=') tra le colonne delle due tabelle coinvolte. L'equi join restituisce solo le righe che hanno un valore corrispondente in entrambe le tabelle.

**Esempi di Algebra Relazionale**

Sala (nome, #posti)

Film (titolo, durata, data)

Proiezione (ID, data, ora, sala\*, film\*)

* Film usciti il Gennaio 2022 =
* In che data sono stati proiettati =

per comodità assegno i film ad una nuova relazione:

* Il nome della sale in cui sono stati proiettati i film del Gennaio 2022 di prima
* Le sale in cui sono stati proiettati i film A ma solo se hanno più di 50 posti (devo accedere a Sala)
* Film con una durata > 100 min
* Sale in cui proiettano i film
* ID della proiezione