Introduzione all'ottimizzazione di query SQL

Geremia Matteo
Antoniazzi Francesco Andrea
Supervisor: Ponzoni Roberto

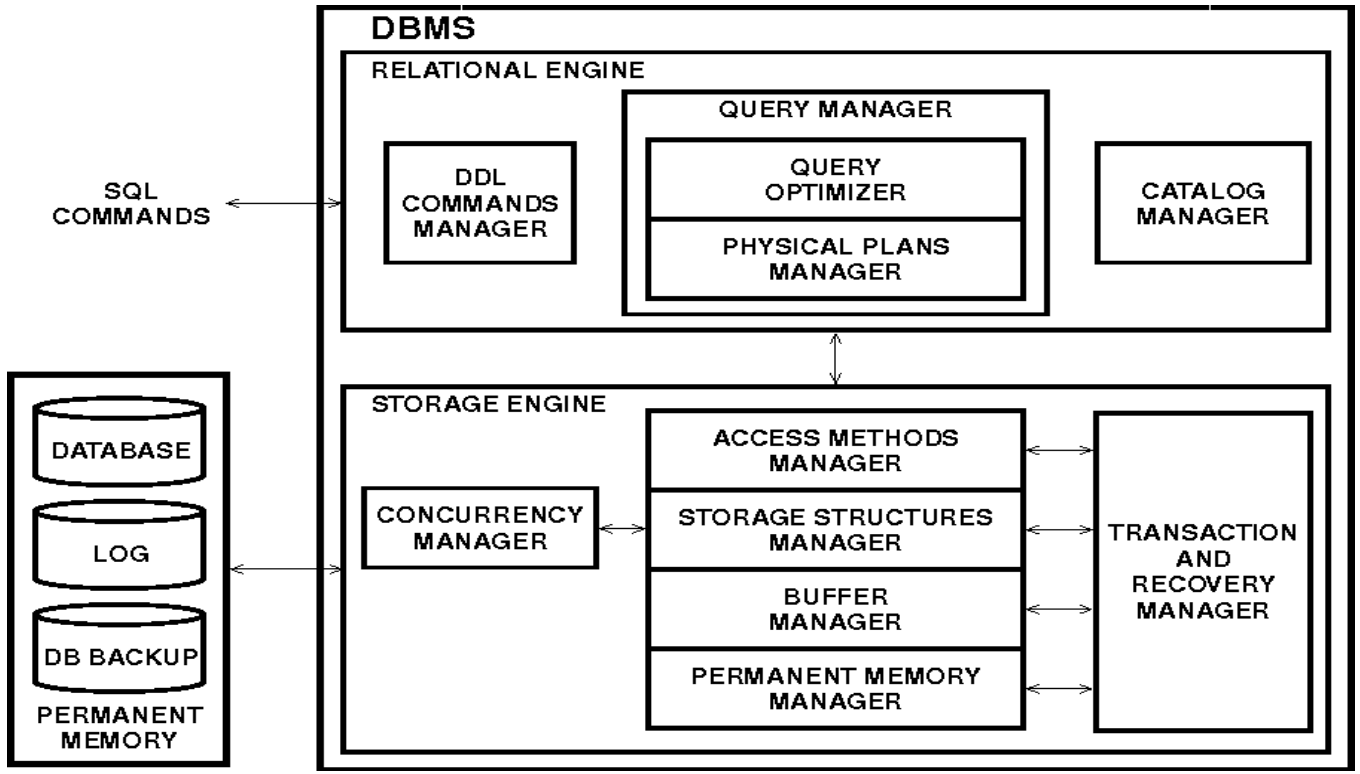
Venezia, 16/11/2023

Indice analitico

- Aspetti principali di DBMS
- Ottimizzazione delle query (SELECT, INSERT)
- Esercizi e quiz!
- Altre tecniche di ottimizzazione (Partizionamento/Read Replica)
- Gotcha e conclusioni
- Riferimenti utili

Che cos’ è un DBMS?

È un **sistema software** progettato per consentire la creazione, la manipolazione e l'interrogazione efficiente di database.



Componenti principali DB

1. Tabelle
2. **Chiavi** (Primary key, Foreign key)
3. Vincoli/**Costraint** (Tipi di dati, Uniqueness, Nullable…)
4. **Viste** (Materializzate e non)
5. **Indici** (Albero, Hash, Indici composti, Geospaziali)
6. Procedure Memorizzate
7. Funzioni
8. Trigger

Transazione SQL

una transazione rappresenta un insieme di operazioni o istruzioni che vengono eseguite come un'unica unità atomica

1.Inizio della Transazione

- Comando: **BEGIN TRANSACTION**
- Descrizione: Segnala l'avvio delle operazioni sul database.

2.Operazioni di Database

- Azioni: **INSERT**, **UPDATE**, **DELETE**, **SELECT**
- Descrizione: Modifica e recupero dei dati.

3.Commit o Rollback

- **COMMIT**: Rende permanenti le modifiche in caso di successo.
- **ROLLBACK**: Annulla le modifiche in caso di errori o problemi.

ACID

ACID è un acronimo che rappresenta quattro principi fondamentali per garantire l'affidabilità e la coerenza nei database relazionali:

1. **Atomicità**: Le operazioni sono indivisibili, e se una transazione fallisce a metà, il database viene ripristinato allo stato precedente.
2. **Coerenza**: Il database deve rimanere in uno stato coerente e valido prima e dopo ogni transazione, rispettando le regole di integrità.
3. **Isolamento**: Le transazioni in corso non devono interferire tra loro, garantendo che sembrino essere eseguite in isolamento.
4. **Durabilità**: Le modifiche apportate da una transazione devono essere permanenti, anche in caso di guasti di sistema.

Questi principi assicurano l'integrità e l'affidabilità dei dati nei database relazionali.

Query Plan

Un query plan è una **rappresentazione dettagliata di come il DBMS intende eseguire una specifica query SQL**.

Questo piano definisce

- la strategia di esecuzione, compresa l'ottimizzazione,
- la sequenza di operazioni
- l'uso di indici

È uno strumento **essenziale** per comprendere come vengono processate le query e identificare problemi di prestazioni.

Indexing

L'indexing è una tecnica utilizzata nei database per migliorare l'efficienza delle operazioni di interrogazione. Si basa sulla creazione di strutture dati specializzate, chiamate indici, che consentono un accesso più rapido e efficiente ai dati all'interno di una tabella.

Indice Singolo:
Basato su una singola colonna per migliorare le query specifiche.
Indice Composito:
Coinvolge più colonne, utile per ottimizzare le query complesse.
Indice Unico:
Assicura l'unicità dei valori, spesso associato a chiavi primarie.

- B-Tree
  - Struttura dati ad albero
  - Operatori ammessi: < <= = >= > e alcuni LIKE
- Hash
  - 32-bit hash code derivato dal valore della colonna indicizzata
  - Operatore ammesso: =

Come scegliere gli indici?

Vantaggi degli Indici

1. Ricerca Rapida
- Accelerano la ricerca dei dati, riducendo il tempo di accesso alle righe desiderate nelle tabelle.

2. Miglioramento delle Performance delle Query
- Le query che coinvolgono colonne indicizzate eseguono più velocemente grazie alla riduzione del numero di operazioni di lettura.

3. Ordinamento Efficiente
- Gli indici facilitano l'ordinamento dei dati, contribuendo a ottenere risultati ordinati senza dover ricorrere a costose operazioni di ordinamento.

4. Join Ottimizzati
- Migliorano le performance delle operazioni di join, specialmente quando coinvolgono colonne indicizzate.

5. Vincoli di Unicità e Chiave Primaria
- Gli indici supportano la definizione di vincoli di unicità e chiavi primarie, garantendo l'integrità dei dati.

Svantaggi degli Indici

1. Overhead di Aggiornamento
- Aumentano il costo di aggiornamento, inserimento e cancellazione dei dati, poiché devono essere mantenuti sincronizzati con le modifiche alla tabella.

2. Spazio su Disco Aggiuntivo
- Ogni indice occupa spazio su disco aggiuntivo, aumentando i requisiti di storage complessivi.

3. Complessità nella Scelta degli Indici Giusti
- La scelta di quali colonne indicizzare richiede una comprensione approfondita delle query e delle operazioni eseguite sulla tabella.

4. Impatto sulle Performance di Scrittura Intensive
- Intensive operazioni di scrittura possono subire rallentamenti a causa della necessità di mantenere gli indici aggiornati.

5. Costi di Manutenzione
- La gestione e la manutenzione degli indici richiedono attenzione, specialmente in ambienti in cui le tabelle subiscono frequenti modifiche.

Per scegliere le colonne su cui creare un indice, va compreso come si intende interrogare la tabella

Va considerata la **selettività** di un indice

ESERCIZI

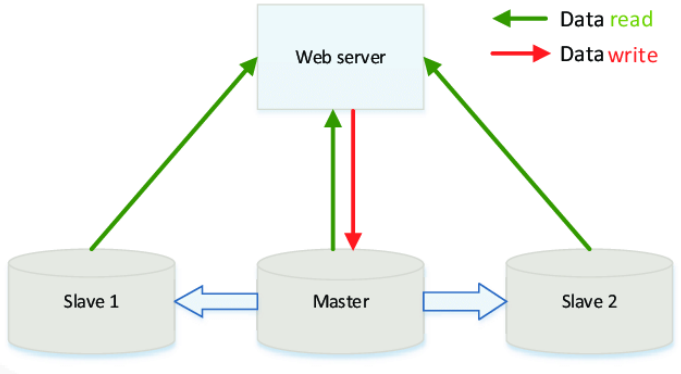
Master-Replica

La Master-Replica Architecture è un modello di distribuzione dei dati in cui esiste un nodo principale (master) e uno o più nodi secondari (replica)

- Master:
  - Contiene il set completo dei dati e gestisce tutte le operazioni di scrittura (inserimento, aggiornamento, cancellazione). Fornisce la fonte autoritativa dei dati.
- Replica:
  - Contiene una copia dei dati presenti sul master.
  - Utilizzata principalmente per operazioni di lettura per distribuire il carico di lavoro e migliorare le prestazioni.

Vantaggi

- **Scalabilità delle Letture**
- **Tolleranza ai Guasti**
- **Backup e Recupero**
- **Geodistribuzione**



Utilizzabile in caso una Eventual Consistency sia accettabile

Partitioning

- Il partitioning in PostgreSQL è una funzionalità che consente di suddividere grandi tabelle in parti più gestibili chiamate partizioni, migliorando le prestazioni e semplificando la gestione dei dati.

1. **Range Partitioning:**

sql

```sql
CREATE TABLE sales (
    sale_id serial PRIMARY KEY,
    sale_date date,
    amount numeric
) PARTITION BY RANGE (sale_date);

-- Definizione della prima partizione
CREATE TABLE sales_january PARTITION OF sales
    FOR VALUES FROM ('2023-01-01') TO ('2023-02-01');

-- Definizione di un'altra partizione
CREATE TABLE sales_february PARTITION OF sales
    FOR VALUES FROM ('2023-02-01') TO ('2023-03-01');
```

```sql
CREATE TABLE sensor_data (
    sensor_id serial PRIMARY KEY,
    sensor_value double precision
) PARTITION BY HASH (sensor_id);

-- Definizione delle partizioni (ad esempio, quattro partizioni)
CREATE TABLE sensor_data_1 PARTITION OF sensor_data FOR VALUES WITH (MODULUS 4, REMAINDER 0);
CREATE TABLE sensor_data_2 PARTITION OF sensor_data FOR VALUES WITH (MODULUS 4, REMAINDER 1);
CREATE TABLE sensor_data_3 PARTITION OF sensor_data FOR VALUES WITH (MODULUS 4, REMAINDER 2);
CREATE TABLE sensor_data_4 PARTITION OF sensor_data FOR VALUES WITH (MODULUS 4, REMAINDER 3);
```

2. **List Partitioning:**

sql

```sql
CREATE TABLE employees (
    emp_id serial PRIMARY KEY,
    emp_name text
) PARTITION BY LIST (emp_name);

-- Definizione della prima partizione
CREATE TABLE employees_a PARTITION OF employees
    FOR VALUES IN ('Alice', 'Alex');

-- Definizione di un'altra partizione
CREATE TABLE employees_b PARTITION OF employees
    FOR VALUES IN ('Bob', 'Bill', 'Barbara');
```

Gotcha

Ho una tabella con decine di milioni di elementi. Mi chiedono di inserire a FE l'informazione aggiornata del numero di elementi. Non serve il numero esatto, bensì una stima verosimile.

Che query scriveresti a DB?

```
SELECT count(*) AS estimate FROM reports;
```

Questa query è lenta su un numero molto elevato di record

```
SELECT reltuples AS estimate FROM pg_class WHERE relname = 'reports';
```

Questa query è veloce perché utilizza le statistiche della tabella, e non i dati associati

Bibliografia

- PostgreSQL Doc - Using Explain
- PostgreSQL Doc - Index Types
- Youtube - Best Practices Working with Billion-row Tables in Databases
- Youtube - don’t use “offset” in your SQL

- Corso Udemy molto interessante
- Buona pasqua!

Grazie per l'attenzione

Geremia Matteo
Antoniazzi Francesco Andrea
Supervisor: Ponzoni Roberto

Venezia, 16/11/2023