IN2090 – Databaser og datamodellering

13 – Indekser og Spørreprosessering

Leif Harald Karlsen
leifhka@ifi.uio.no

Universitetet i Oslo

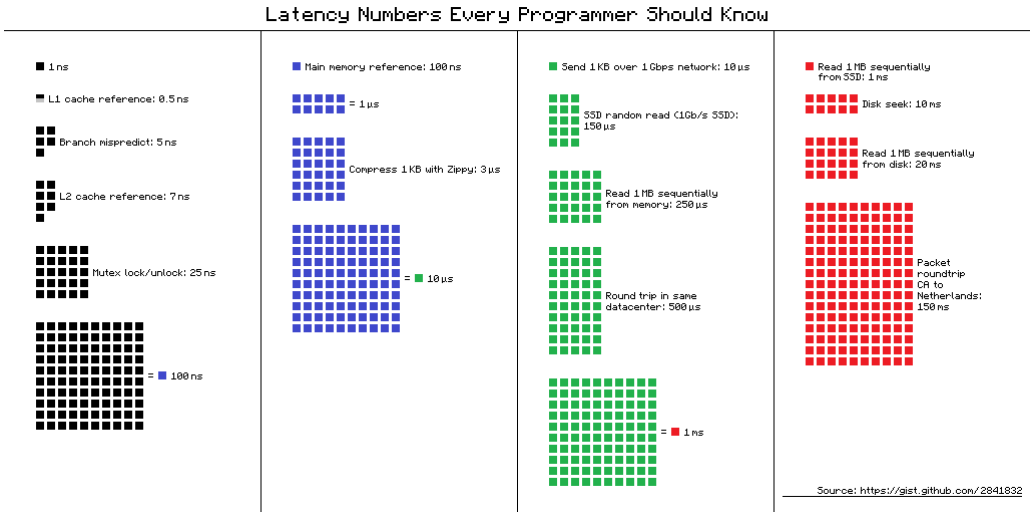
Spørringer og kompleksitet

- Hvordan utfører en database et oppslag på en bestem verdi?
- Eller en join mellom to tabeller?
- Begge disse problemene er egentlig et søk etter bestemte verdier i en kolonne
- For at databasen skal kunne utføre disse operasjonene effektivt (spesielt over veldig store tabeller) trenger vi datastrukturer som gjør søket mer effektivt
- Slike datastrukturer heter indeksstrukturer

Indeksstrukturer

- En indeksstruktur er en datastruktur som lar databasen hurtig finne bestemte rader i en tabell, basert på verdiene i en (eller flere) kolonner
- Husk at databaser lagrer data på disk, ikke i minne (RAM)
- Databaseindekser skiller seg litt fra andre datastrukturer fordi de ikke lagres i minne, men på disk

Lese fra harddisk

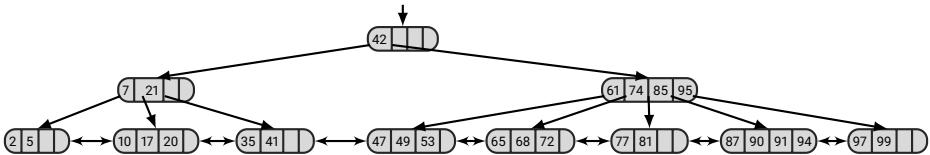


Hentet fra `https://gist.github.com/hellerbarde/2843375`

Lese fra harddisk

- Å lese fra disk tar ca. 10,000 ganger lengre tid enn fra RAM (avhenig av disktype og minnetype)
- Indeksstrukturer i databaser er derfor optimert for å utføre så få diskoppslag som mulig
- Har to hovedtyper indekser: Hash-baserte og tre-baserte

B-tre-indekser



- Trestruktur hvor hver node kan ha mange barn
- Nodene har samme størrelse som en disk-blokk
- Minimerer antall oppslag på disk
- Hver verdi i løvnodene har pekere til dens tilhørende rad i den tilhørende tabellen
- Kan utføre effektive oppslag på konkrete verdier
- Samt effektive intervall søk

Hash-indekser

- En hash-indeks bruker en hash-funksjon for å oversette en verdi til en minneadresse
- På minneadressen ligger så en liste med pekere til rader som har denne verdien
- Hash-indekser er mer effektive på oppslag på konkrete verdier
- Men kan ikke brukes for intervaller (må da gjøre ett oppslag for hver mulige verdi i intervallet)

Andre indeksstrukturer

- Det finnes mange andre indeksstrukturer
- Ulike strukturer er tilpassed ulike datatyper
- Egne strukturer for f.eks.:
  - Søk i tekst
  - Romlige data og koordinater i høyere dimensjoner
  - Sammensatte strukturer (JSON, XML, osv.)

Nøkler og indekser

- Når man markerer en kolonne med `PRIMARY KEY` blir det automatisk opprettet en B-tre-indeks på denne kolonnen
- Joins over primærnøkler er derfor alltid relativt effektive
- Men, det kan hende man ønsker å gjøre søk, oppslag eller joins over kolonner som ikke er primærnøkler
- Vi må da lage indeksene selv

Lage indekser med SQL

- For å lade en indeks på en kolonne trenger man bare å skrive

  ```
  CREATE INDEX <index_name> ON <table>(<columns>);
  ```

  hvor `<index_name>` er navnet på indeksen, `<table>` er et tabellnavn og `<columns>` er en liste med kolonner man ønsker å indeksere
- Databasen lager da en passende indeks (typisk B-tre)
- F.eks.:

  ```
  CREATE INDEX price_index ON products(unit_price);
  ```

- Merk: Dersom man lister opp flere kolonner blir indeksen over alle kolonnene samtidig
- Databasen finner selv ut når indeksen bør brukes

Indeks-demonstrasjon

```
DROP TABLE IF EXISTS t1;
DROP TABLE IF EXISTS t2;

CREATE TABLE t1(id int);
INSERT INTO t1
SELECT n*random() -- Genererer 10 mill. tilfeldige tall
FROM generate_series(1, 10000000) AS x(n);

CREATE TABLE t2 AS (SELECT * FROM t1); -- t2 inneholder samme data som t1

CREATE INDEX t1_ind ON t1(id); -- Lager index på t1

-- Følgende gjør at psql printer ut hvor lang tid spørringer tar
\timing on

SELECT * FROM t1 WHERE id = 100; --> Time:   0.792 ms
SELECT * FROM t2 WHERE id = 100; --> Time: 303.022 ms
```

Takk for nå!

Neste video handler om spørreprosessering.