Adatbázisok - 9. előadás

Horváth Árpád <horvath.arpad@amk.uni-obuda.hu>

Óbudai Egyetem
Alba Regia Műszaki Kar (AMK)
Székesfehérvár

2015. október 15.

Vázlat

Indexek

Az index fogalma

- Egy feltételes SELECT utasítás végrehajtásakor az adatbázis-kezelő a teljes táblában keres, sorról-sorra összehasonlítva az értékeket a feltétellel. Egy tábla esetén, amelyben több millió rekord található, egy ilyen utasítás végrehajtása nem hatékony. Ezért rendelkezésre áll egy indexelő mechanizmus, amellyel megadott oszlopokat indexelhetünk. Ha indexelünk egy oszlopot és a feltételes utasításunkat (amely az indexelt oszlopban keres) végrehajtjuk, akkor az adatbázis-kezelő a tábla helyett az indextáblában hajt végre keresést.

Az index felépítése

- Az indexet többféleképpen implementálhatják (a gyakorlatban bináris fát, egyéb kereső fákat vagy hash táblákat alkalmaznak az indexek tárolására), a legegyszerűbben azonban úgy szemléltethetjük, mint egy rendezett listát amelynek értékei hivatkoznak a tábla azon soraira amelyben az indexelt oszlop (vagy oszlopok) értékei megtalálhatók.
- Az indexek segítségével **bináris keresés** hajtható végre az adathalmazon.
- A PostgreSQL automatikusan létrehoz indexet a `PRIMARY KEY` és a `UNIQUE` kényszerek alkalmazásakor.

Index alkalmazásának indokai

- Általában az indexek alkalmazása indokolt azoknál az oszlopoknál, amelyeket gyakran:
  - lekérdeznek (WHERE),
  - rendeznek (ORDER BY),
  - csoportosítanak (GROUP BY),
  - használnak az összekapcsolás műveleteiben (JOIN),
  - használnak számításokra (MIN(), MAX(),...)

Esetek, amikor az indexelés nem indokolt

- Általában index alkalmazása nem javasolt azon oszlopokon, amelyek
  - csak kevés egyedi értéket tartalmaznak vagyis csupán néhány értéket tartalmaznak, amelyek gyakran ismétlődnek (pl. nem (ffi, nő), családi állapot,...)
  - ritkán szerepelnek lekérdezésekben
  - kevés rekordot tartalmazó táblának a részei

Példa: Index készítése

```
1  CREATE TABLE teszt (
2    a_oszlop int,
3    b_oszlop int,
4    c_oszlop varchar
5  );
```

Index készítése

```
1  CREATE INDEX a_oszlop_idx ON teszt (a_oszlop);
```

Ha gyakoriak az ilyen lekérdezések:

```
1  SELECT name FROM teszt
2  WHERE a_oszlop = 8 AND b_oszlop = 8;
```

célszerű lehet mindkét oszlop szerint együtt indexelni:

```
1  CREATE INDEX a_b_idx ON teszt (a_oszlop, b_oszlop);
```

Vázlat

Kurzorok (cursor)

A kurzor fogalma

- Amikor végrehajtunk egy SELECT utasítást, akkor egyszerre kapunk meg minden sort. Ez nem mindig megfelelő, előfordulhat például, hogy egy adott sor visszakapott oszlopértékei alapján valamilyen műveletet szeretnénk végezni. Ehhez egy kurzort (cursor, sormutató) kell használnunk, amellyel az adatbázisból kinyert sorokat egyenként dolgozhatjuk fel. A kurzor segítségével végiglépkedhetünk az adott SELECT utasítás által visszaadott sorokon.
- Használata az alkalmazásfejlesztésben lényeges, mert egy perzisztens (tartós) kapcsolat épül ki az adatbázis szerver és a kliens alkalmazás között, így egy eredményhalmaz adatait pl. sorról sorra be lehet olvasni az alkalmazásba, majd további feldolgozást végezni azokon.

Kurzor használatának lépései

- Kurzor deklarálása (DECLARE)
- Sor kiolvasása (FETCH)
- Kurzor mozgatása (MOVE)
- További olvasás és mozgatás
- Kurzor lezárása (CLOSE)
- A lépéseket tranzakcióba foglalhatjuk.

Példa: Kurzor használata

```
1  BEGIN; -- tranzakcio inditasa
2  DECLARE osszes CURSOR  -- deklaralas
3  FOR SELECT * FROM autok;
4
5  FETCH 2 FROM osszes; -- 2 sor kiolvasasa
6  MOVE FORWARD 4 IN osszes; -- mozgatas elore 4 sorral
7  FETCH NEXT FROM osszes; -- a kovetkezo sor kiolvasasa
8
9  CLOSE osszes; -- kurzor lezarasa
10 COMMIT; -- tranzakcio lezarasa
```

Vázlat

Triggerek (trigger function)

A trigger fogalma

- A trigger (kioldó) olyan különleges tárolt függvény, amelyet az adatbázis-kezelő automatikusan futtat, amikor egy meghatározott INSERT, UPDATE vagy DELETE utasítást egy bizonyos adatbázistáblán végrehajtunk.
- A triggerek igen hasznosak például akkor, ha egy tábla oszlopértékeinek változásait szeretnénk ellenőrizni. Olyan hibák akadályozhatók meg, amelyeket a CHECK constraint használatával nem tudunk ellenőrízni.
- Egy trigger az INSERT, UPDATE vagy DELETE utasítások helyett, előtt vagy után is elindulhat.

Vázlat

Az SQL kiterjesztése

Az SQL, mint deklaratív nyelv

- Az SQL az ún. deklaratív programozási nyelvek (deklaratív paradigma) csoportjába tartozik, mert:
  - nem azt határozzuk meg, hogy hogyan kell egy számítást végrehajtani
  - azt adjuk meg, hogy **mit** kell végrehajtania az adatbázis-kezelőnek.
- Az SQL nem rendelkezik vezérlési szerkezetekkel.

Példa: Autók adatait tartalmazó adathalmaz

- Feladat adatok kiolvasása az adathalmazból

rendszam	tipus
ABC-123	Ford
DEF-456	Mazda
GHI-789	Renault

Példa: Python megoldás - 1

```
1  autok = {"ABC-123": "Ford", "DEF-456": "Mazda",
2           "GHI-789": "Renault"}
3
4  for rendszam in autok:
5    print(rendszam, autok[rendszam])
```

- Szkript futtatása: `python autok.py`
- Eredmény:

```
ABC-123 Ford
DEF-456 Mazda
GHI-789 Renault
```

Példa: Python megoldás - 2

```
1  autok = [{"rendszam": "ABC-123", "tipus": "Ford"},
2           {"rendszam": "DEF-456", "tipus":"Mazda"},
3           {"rendszam": "GHI-789", "tipus": "Renault"}]
4
5  for auto in autok:
6    for kulcs in auto:
7      print(kulcs, ":", auto[kulcs], end=" ")
8    print()
```

- Szkript futtatása: `python autok2.py`
- Eredmény:

```
rendszam : ABC-123 tipus : Ford
rendszam : DEF-456 tipus : Mazda
rendszam : GHI-789 tipus : Renault
```

Példa: SQL megoldás

```
1 SELECT rendszam, tipus
2 FROM autok;
```

SQL kiterjesztése nyelvi elemekkel

- Az adatbázis-kezelők rendszerint biztosítanak további nyelvi elemeket az SQL kiegészítésére (pl. vezérlési szerkezetek, változók,. . .)
- PostgreSQL-ben a következő nyelvek használhatók:
  - C nyelvben megírt kiterjesztések (extension), amelyek betöltődnek a szerver indításakor
  - PL/pgSQL - beépített
  - Pl/Tcl, PL/Perl, PL/Python - az operációs rendszeren telepített Tcl, Perl vagy Python nyelvi környezet használatával

Vázlat

PL/Python

PL/Python

- Triggerek létrehozása
- Függvények létrehozása
- Vezérlési szerkezetek, változók, Python modulok használata
- A PL/Python a Python 2. és 3. kiadását is támogatja

PL/Python telepítése adatbázisba

- Az extension elkészítését követően használhatjuk a PL/Pythont

```
psql -d teszt_db
```

```
1  -- Python 2
2  CREATE EXTENSION plpython2u;
3  -- vagy Python 3
4  CREATE EXTENSION plpython3u;
5  -- vagy az operacios rendszerben az  alapertelmezett
6  -- Pythont hasznalva:
7  CREATE EXTENSION plpythonu;
```

Függvény készítése

- Függvény felépítése:
  - Utasítás, pl.: `CREATE FUNCTION fuggveny_nev`
  - Argumentumok vesszővel elválasztva és típussal megadva, pl.: `(a integer, b integer)`
  - Visszatérési érték típusa, pl.: `RETURNS integer`
  - Nyelvi blokk kezdete, pl.: `AS $$`
  - A függvény törzse
  - A nyelvi blokk és a fv. lezárása, pl.: `$$ LANGUAGE plpythonu;`

Példa: Egyszerű függvény (1)

```
1  CREATE FUNCTION pymax (a integer, b integer)
2    RETURNS integer
3  AS $$
4    if (a is None) or (b is None):
5      return None
6    if a > b:
7      return a
8    return b
9  $$ LANGUAGE plpythonu;
```

Példa: Egyszerű függvény (2)

- Függvény használata

```
1  SELECT pymax(4, 5);
```

Példa: Összetett függvény (1)

```
1  CREATE TABLE alkalmazottak (
2    nev text,
3    fizetes integer,
4    kor integer
5  );
6
7  CREATE FUNCTION ber_kompenzacio (alk alkalmazottak)
8    RETURNS boolean
9  AS $$
10   if alk["fizetes"] > 300000:
11     return False
12   if (alk["kor"] < 30) and (alk["fizetes"] > 200000):
13     return False
14   return True
```

Példa: Összetett függvény (2)

```
1  INSERT INTO alkalmazottak
2  VALUES
3  ('Janos', 320000, 35),
4  ('Peter', 250000, 25),
5  ('Bela', 150000, 22),
6  ('Istvan', 270000, 32);
7
8  SELECT nev, ber_kompenzacio(alkalmazottak.*)
9  FROM alkalmazottak;
```

Eredmény:

```
Janos  f
Peter  f
Bela   t
```

Vázlat

PostGIS műveletek, lásd: m.postgis_muveletek.pdf

Függvények használatának jelölése

- Visszatérési érték típusa, pl.: geometry
- Függvény neve, pl.: st_geomfromtext
- Argumentumok zárójelben, vesszővel elválasztva:
  - argumentum típusa, pl.: text, integer
  - argumentum, pl.: WKT, srid, geom
- Néhány függvénynél némelyik argumentum elhagyható, ekkor több formula kerül megadásra, pl. az st_geomfromtext() függvény egy vagy kettő argumentummal hívható meg

Példa jelölésre

```
geometry ST_GeomFromText(text WKT);
geometry ST_GeomFromText(text WKT, integer srid);
```

```
1 SELECT st_geomfromtext(
2 'LINESTRING (100 100, 20 180, 180 180)');
3 -- GPS koordinata, SRID = 4326
4 SELECT st_geomfromtext(
5 'POINT(18.409 47.1903)', 4326);
```