Adatvezérelt Alkalmazások Fejlesztése

Minta ZH

KIDOLGOZÁS

1. Lekérdezés optimalizálás során miképp valósítják meg a join operátort? Az egyes megvalósítások hogyan működnek, mekkora az I/O költségük? (3 pont)

Beágyazott ciklusok (Nested Loops Join)

- Egymásba ágyazott kettős for ciklus
- I/O költség
  - Nyalábolt esetben: O(B(R)·B(S))
  - Nem nyalábolt esetben: O(T(R)·T(S))
- Tetszőleges méretű táblákra működik
  - Nagy méret esetén: a két tábla egy- egy blokkját tartja
  - memóriában
  - Kis méret: a táblát bent lehet tartani

Hash alapú összekapcsolás (Hash Join)

- Első menetben
  - Kisebb (R) reláció beolvasása
  - Vödrös hash építése a memóriában
  - Kulcs a join operátorban szereplő oszlop
- Második menet
  - A nagyobbik (S) reláció beolvasása
  - Kapcsolódó rekordok keresése a vödrös hashben
- I/O költség
  - O(B(R)+S(R))

Rendezés összefésülés (Sort Merge Join)

- Mindkét relációt beolvassa a memóriába
- Rendezi az összekapcsolási kulcs szerint
- A két rendezett listát összefésüli
  - Listák közös bejárása
  - Egyszerű algoritmus
- Kis méretű relációk esetén
- Index a rendezés miatt
- I/O költség
  - O(B(R)+S(R))

2. Objektum relációs leképezés során miképp lehet az öröklési hierarchiát leképezni relációs adatmodellre? (4 pont)

Hierarchia leképezése egy közös táblába

Egyszerű hierarchiák leképezéséhez.

Megvalósítás

- az objektum minden attribútumát (örököltekkel) felsoroljuk
- objektumtípusok (osztályok) tárolása:
  - minden típushoz egyedi azonosító VAGY
  - IsClass1, IsClass2, … oszlopok
- egyedi példányazonosító tárolása
- bővítés kezelése: új attribútumok felvitele

Jellemzők

- ☺ egyszerű
- ☺ könnyű új osztályt vinni a hierarchiába
- ☺ könnyen követhető, melyik objektum milyen osztályok példánya
- ☹ helypazarló
- ☹ egy osztály megváltozása az egész tárolást változtatja
- ☹ nehezen áttekinthető

Minden valós osztály leképezése saját táblába

Ritkán változó struktúrák leképezéséhez.

Megvalósítás

- minden osztálynak saját tábla
- az osztály minden attribútumának eltárolása
- egyedi példányazonosító tárolása
- ha változik egy attribútum, csak a hierarchiaszint mentén kell végigvinni

Jellemzők

- ☺ átlátható
- ☺ gyors adatelérés
- ☺ jól illeszkedik az objektummodellhez
- ☹ osztály módosítása → hierarchiaszintek módosítása
- ☹ több szerepet is betöltő példányok kezelése nehézkes

Minden osztály leképezése saját táblába

Komplex öröklési hierarchia és gyakran változó struktúrák leképezéséhez.

Megvalósítás

- a táblák követik az osztályhierarchiát
- szülő-gyerek viszony leképezése külső kulcsokkal
- egyedi példányazonosító tárolása

Jellemzők

- ☺ osztályok struktúrája könnyen módosítható
- ☹ összetett DB-séma
- ☹ egy példány adatai több táblában vannak (öröklés miatt) → összetett lekérdezés

Osztályok és hierarchiaszintek általános leképezése

Komplex alkalmazásokban, kis mennyiségű adat mellett, ahol akár futási időben is minden változhat.

Megvalósítás

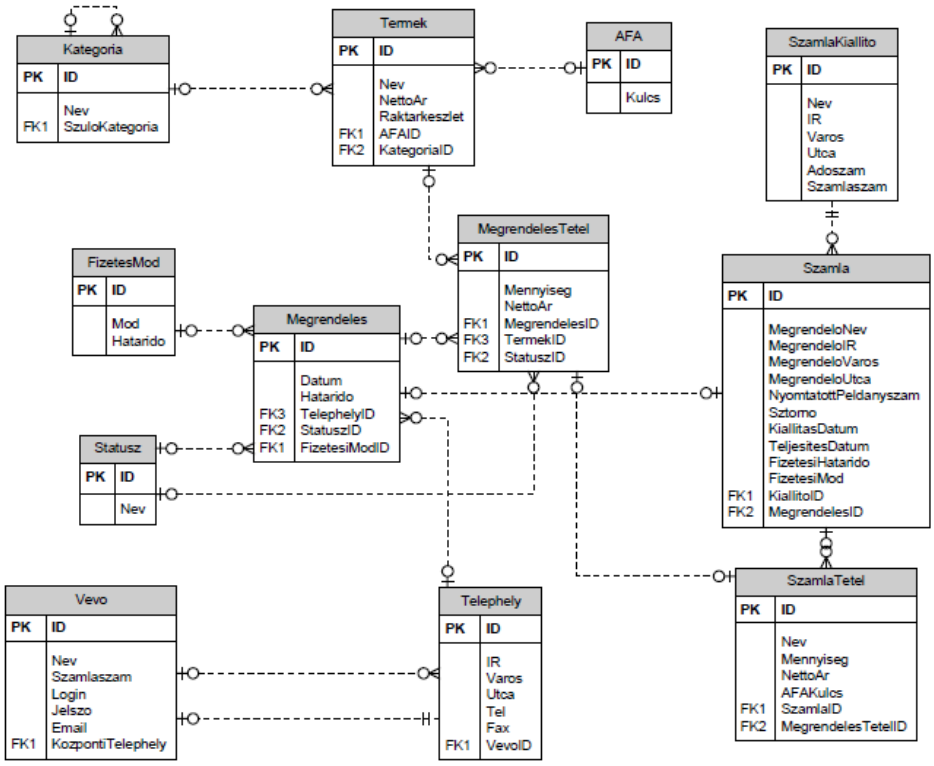
- metaadat-központú
- általános séma: tetszőleges hierarchia leírható – hierarchia → metaadat, példányok → attribútumok
- class, attribute, attributeType, value, inheritance, stb… táblák

Jellemzők

- ☺ perzisztencia-kezelő rendszerekhez illeszkedik
- ☺ flexibilis, bármi leírható benne
- ☹ nehéz összeszedni egy példány adatait
- ☹ sok adat esetén nem hatékony

3. Mire szolgálnak a Dependency Property-k a WPF-ben? (3 pont)

Ilyenről volt szó?



4. Adott a fenti adatmodell. A rendszer lassú működése miatt szükséges a megrendelés táblába felvenni egy új oszlopot mely a megrendeléstételekből számítható végösszeget tartalmazza. Készítsd el a szükséges módosításokat végrehajtó SQL szkriptet Oracle szerveren, valamint a folyamatos karbantartáshoz szükséges triggereket! (5 pont)

```
-- felveszem az új oszlopot a megrendelés táblába
alter table Megrendeles add vegosszeg float;

-- létrehozom a triggert
create or replace trigger VegosszegKarbantartasTrigger
  after update or delete on MegrendelesTetel
  for each row
declare
  -- local variables here
begin

  update megrendeles
  set vegosszeg = vegosszeg + nvl(:new.NettoAr * :new.Mennyiseg, 0)
                            - nvl(:old.NettoAr * :old.Mennyiseg, 0)

end VegosszegKarbantartasTrigger;
```

5. **Tekintsük a fenti adatmodell LinQ leképzését! Készíts olyan C# nyelvű, LinQ lekérdezést tartalmazó kódrészletet, mely kilistázza a 25000 Ft-nál nagyobb összegű megrendeléseket és ezt követően ezen megrendelések tételeire 10% árkedvezményt ad. (5 pont)**

```
string connStrt = "Data Source=mezga;Initial Catalog=NEPTUN;User
  ID=NEPTUN;Password=NEPTUN";

using (DataContext dc = new DataContext(connStrt))
{
  var qDragaMegrendelesek = from m in dc.Megrendeles
                            where m.Vegosszeg > 25000
                            select m;

  foreach (var m in qDragaMegrendelesek)
  {
      m.Vegosszeg = m.Vegosszeg * 0.9;
  }

  dc.SubmitChanges();
}
```