Andmebaaside teooria kursuse kodutöö

Aleksei Gorny
Irina Ivanova
Ilja Kuzovkin

June 1, 2009

Andmebaasi vajalikkus

Enamik vahetevahel parte toitma sattuvaid inimesi eeldab vaikimisi, et nende tegevus leiab lindudelt heakskiitu. Praktikas on aga hobikorraline linnuarmastaja seni olnud võimetu söödeteva poole heaolu tagama, kuna see nõuab laialdasi teadmisi partide ajas muutuvatest individuaaleelistustest. Veelgi enam, tihti võib partide toiduvajadus jääda üldsegi täitmata, kuna info lindude asukoha kohta pole avalikult kättesaadav.

Ülesande püstitus

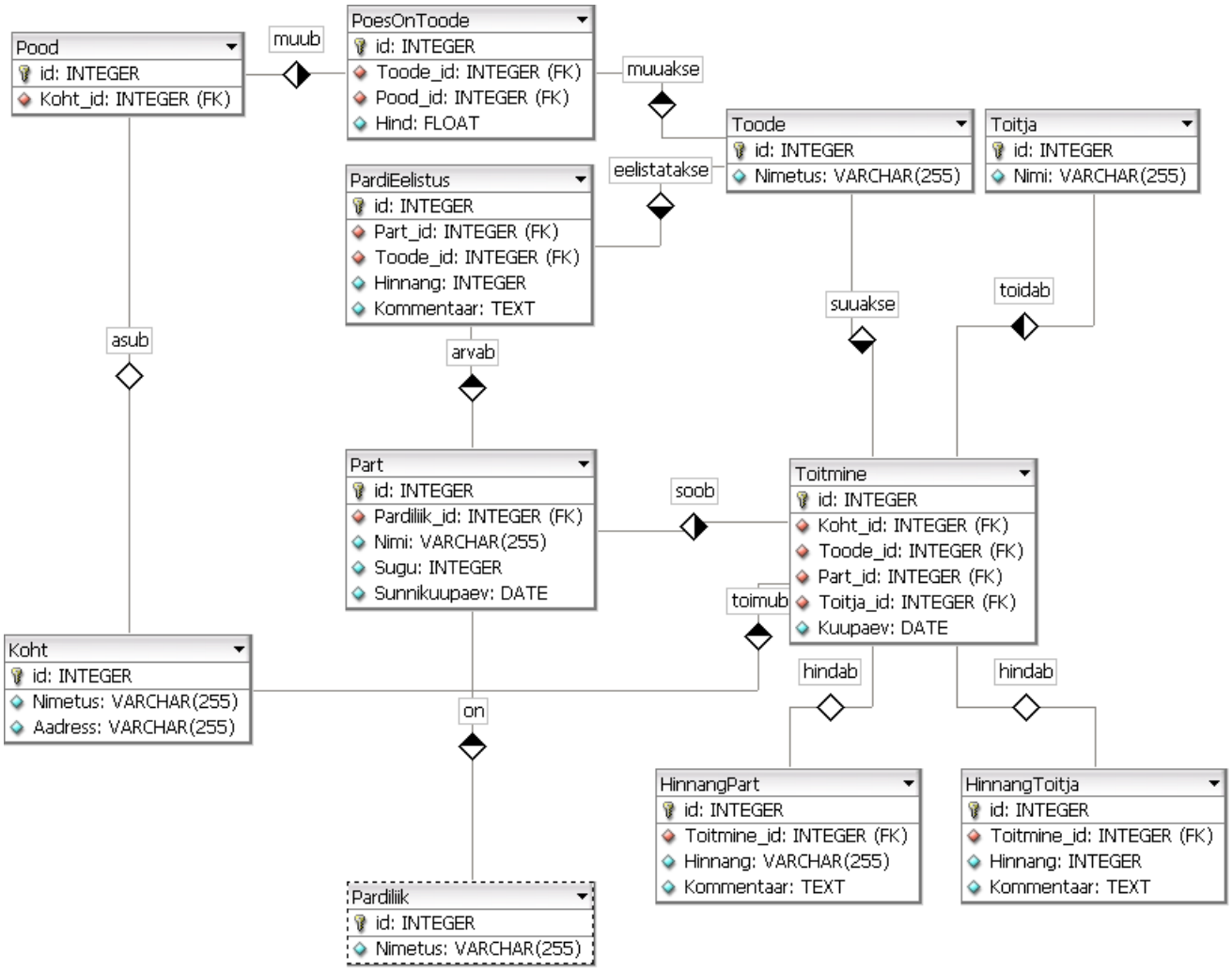
Meie eesmärgiks on projekteerida andmebaas haldamaks informatsiooni toitmisprotsessiga seotud objektide kohta ning võimaldada osapooltel vahetada tagasisidet. Andmebaas sisaldaks andmeid partidele söögiks sobivate toodete kättesaadavuse kohta, partide individuaalhinnguid proovitud toodetele, lindude kogunemis-kohti, toitjate arvamust konkreetsete partide kohta jm. Selle kasutajateks saaksid lisaks mainitud osapooltele olla ka teised huvilised, näiteks lindude käitumistrendide kohta üldistusi teha soovivad ornitoloogid. Siiski on loodav andmebaas eelkõige mõeldud eespool selgitatud kitsaskohtade elimineerimiseks.

Andmemudel

Mõistekaart, märgitud on ainult olemite elementaar-atribuudid:

Toitja	Linde toita sooviv füüsiline või juriidiline isik
Nimi	
Part	Toitmisest huvitatud partlaste sugukonna esindaja
Nimi	
Sugu	
Sünnikuupäev	
PardiLiik	Pardiliigi täpne nimetus bioloogilise taksonoomia järgi
Nimetus	
Koht	Üldistus määramaks toitmise & poodide asukohti
Nimetus	
Aadress	
Toode	Partide poolt söödav objekt
Nimetus	

Toitmine	Kirjeldab ühe konkreetse pardi konkreetse isiku poolt toitmise sündmust
Kuupäev	
HinnangPart	Pardi hinnang konkreetsele toitmissündmusele, milles ta osales
Hinnang	
Kommentaar	
HinnangToitja	Isiku hinnang konkreetsele toitmissündmusele, milles ta osales
Hinnang	
Kommentaar	
PardiEelistus	Pardi arvamus konkreetsest tootest
Hinnang	
Kommentaar	
Pood	Asutus, millest on võimalik saada partidele meelepäraseid tooteid
PoesOnToode	Määrab konkreetse toote olemasolu määratud poes
Hind	



Relatsiooniline esitus

Järelliitega '-id' on ära märgitud välisvõtmed. Iga relatsiooni võtmeks on atribuut 'id'.

Toitja (id, nimi)
Part (id, nimi, sugu, sünnikuupäev, pardiliik-id)
PardiLiik (id, nimetus)
Koht (id, nimetus, aadress)
Toode (id, nimetus)
Toitmine (id, kuupäev, koht-id, pagaritoode-id, part-id, toitja-id)
HinnangPart (id, hinnang, kommentaar, toitmine-id)
HinnangToitja (id, hinnang, kommentaar, toitmine-id)
PardiEelistus (id, hinnang, kommentaar, part-id, pagaritoode-id)
Pood (id, koht-id)
PoesOnToode (id, hind, pagaritoode-id, pood-id)

Kolmas normaalkuju

Igas relatsioonis R on ainsateks mitte-triviaalseteks funktsionaalseteks sõltuvusteks sekundaartunnuste sõltuvused võtmest.

$$\mathrm{id} \mapsto R \backslash \{\mathrm{id}\}$$

Kolmanda normaalkuju definitsioonist on näha, et siis on relatsioonidel normaalkujul olemiseks vajalikud tingimused täidetud.

DBDesigneri väljundskript andmebaasi loomiseks

```
CREATE TABLE HinnangPart (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Toitmine_id INTEGER UNSIGNED NOT NULL,
  Hinnang VARCHAR(255) NULL,
  Kommentaar TEXT NULL,
  PRIMARY KEY(id)
);

CREATE TABLE HinnangToitja (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Toitmine_id INTEGER UNSIGNED NOT NULL,
  Hinnang INTEGER UNSIGNED NULL,
  Kommentaar TEXT NULL,
  PRIMARY KEY(id)
);

CREATE TABLE Koht (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Nimetus VARCHAR(255) NULL,
  Aadress VARCHAR(255) NULL,
  PRIMARY KEY(id)
);

CREATE TABLE Pagaritoode (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Nimetus VARCHAR(255) NULL,
  PRIMARY KEY(id)
);

CREATE TABLE PardiEelistus (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Part_id INTEGER UNSIGNED NOT NULL,
  Pagaritoode_id INTEGER UNSIGNED NOT NULL,
  Hinnang INTEGER UNSIGNED NULL,
  Kommentaar TEXT NULL,
  PRIMARY KEY(id)
);

CREATE TABLE Pardiliik (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Nimetus VARCHAR(255) NULL,
  PRIMARY KEY(id)
);

CREATE TABLE Part (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
```

```
  Pardiliik_id INTEGER UNSIGNED NOT NULL,
  Nimi VARCHAR(255) NULL,
  Sugu INTEGER UNSIGNED NULL,
  Sunnikuupaev DATE NULL,
  PRIMARY KEY(id)
);

CREATE TABLE PoesOnToode (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Pagaritoode_id INTEGER UNSIGNED NOT NULL,
  Pood_id INTEGER UNSIGNED NOT NULL,
  Hind FLOAT NULL,
  PRIMARY KEY(id)
);

CREATE TABLE Pood (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Koht_id INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(id)
);

CREATE TABLE Toitja (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Nimi VARCHAR(255) NULL,
  PRIMARY KEY(id)
);

CREATE TABLE Toitmine (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Koht_id INTEGER UNSIGNED NOT NULL,
  Pagaritoode_id INTEGER UNSIGNED NOT NULL,
  Part_id INTEGER UNSIGNED NOT NULL,
  Toitja_id INTEGER UNSIGNED NOT NULL,
  Kuupaev DATE NULL,
  PRIMARY KEY(id)
);
```

SQL päringud

1) Kõige populaarsem toode selle aasta maikuus.

```
SELECT
  Toode.Nimetus
FROM
  Toode, Toitmine, HinnangPart
WHERE
  Toitmine.Toode_id = Toode.id AND
  HinnangPart.Toitmine_id = Toitmine.id AND
  Toitmine.Kuupaev BETWEEN "2009-05-01" AND "2009-05-30"
GROUP BY Nimetus
ORDER BY sum(HinnangPart.Hinnang) / count(HinnangPart.Hinnang) DESC
LIMIT 1
```

Väljund:

	Nimetus
1	Ene Sai

2) Pood, mille tooted on partidelt pälvinud kõrgeimad hinnagud ning selle keskmine hinnang.

```
SELECT
  Koht.Nimetus, sum(PardiEelistus.Hinnang) / count(PardiEelistus.Hinnang) as KeskHinnang
FROM
  Koht, Pood, PardiEelistus, Toode, PoesOnToode
WHERE
  Pood.Koht_id = Koht.id AND
  PoesOnToode.Pood_id = Pood.id AND
  PoesOnToode.Toode_id = Toode.id AND
  PardiEelistus.Toode_id = Toode.id
GROUP BY Pood.id
ORDER BY sum(PardiEelistus.Hinnang) / count(PardiEelistus.Hinnang) DESC
LIMIT 1
```

Väljund:

Nimetus	KeskHinnang
Selver	6.6667

3) Parim päev toitmiseks: võimalikult palju parte ning võimalikult kõrged hinnangud.

```
SELECT t1.Kuupaev FROM
(
  (
    SELECT
      Toitmine.Kuupaev, Toitmine.id, count(*) as parte
    FROM
      Toitmine
    GROUP BY Toitmine.Kuupaev
    ORDER BY parte DESC
  ) AS t1
  JOIN
  (
  SELECT
    Toitmine.id, (sum(HinnangToitja.Hinnang) + sum(HinnangPart.Hinnang)) /
      (count(HinnangToitja.Hinnang) + count(HinnangPart.Hinnang) ) as k
  FROM
    HinnangToitja, HinnangPart, Toitmine
  WHERE
    HinnangToitja.Toitmine_id = Toitmine.id OR
    HinnangPart.Toitmine_id = Toitmine.id
  GROUP BY
    Toitmine.id
  ORDER BY k DESC
  ) AS t2
) WHERE t1.id = t2.id
LIMIT 1
```

Väjund:

	Kuupaev
1	2009- 05-19

4) Kaks suurima maitseerinevusega pardiliiki

Tööpõhimõte:

- Loome ajutise tabeli, mis sisaldab iga pardiliigi ja toote kohta liigi keskmist hinnangut antud tootele.
- Liigume kolmekordse tsükliga üle toodete ja pardiliigi-paaride. Loome teise ajutise tabeli, kuhu salvestame iga pardiliigi-paari ja toote korral liikide hinnanguerinevuse absoluut- väärtuse.
- Loome kolmanda ajutise tabeli, kuhu salvestame iga pardiliigi-paari jaoks nende hinnanguerinevuste summa
- Valime viimasest tabeli suurima erinevusnumbriga paari

```
delimiter ||
CREATE PROCEDURE erinevadMaitsed()
  BEGIN
    SET @maximum = (SELECT id FROM Pardiliik ORDER BY id DESC LIMIT 1);
    SET @counter = 1;
    CREATE TABLE temp(Pardiliik_id int, Toode_id int, Nimetus char(45), Hinnang float);
    label: LOOP
      IF (@counter < @maximum) THEN
        INSERT INTO temp (Pardiliik_id, Toode_id, Nimetus, Hinnang)
          SELECT Part.Pardiliik_id, Toode.id, Toode.Nimetus, sum(PardiEelistus.Hinnang) /
            count(PardiEelistus.Hinnang) AS Hinnang
          FROM Part,Toode,PardiEelistus
          WHERE PardiEelistus.Part_id = Part.id AND
          PardiEelistus.Toode_id = Toode.id AND
          Part.Pardiliik_id = @counter
          GROUP BY Toode.id;
        SET @counter = @counter + 1;
        ITERATE label;
      END IF;
      LEAVE label;
    END LOOP label;
    CREATE TABLE temp2 (Toode_id int, Pardiliik1_id int, Pardiliik2_id int, diff float);
    SET @maxp = (SELECT max(Pardiliik_id) FROM temp);
    SET @maxt = (SELECT max(Toode_id) FROM temp);
    SET @tcounter = 1;
    SET @p1counter = 1;
    SET @p2counter = @p1counter + 1;
    tooded: LOOP
      IF @tcounter <= @maxt THEN
        SET @p1counter = 1;
        liiged1: LOOP
          IF @p1counter <= @maxp THEN
            SET @p2counter = @p1counter + 1;
            liiged2: LOOP
              IF @p2counter <= @maxp THEN
```

```
            SET @diff =
              (SELECT Hinnang FROM temp WHERE Toode_id = @tcounter AND
                Pardiliik_id = @p1counter) -
              (SELECT Hinnang FROM temp WHERE Toode_id = @tcounter AND
                Pardiliik_id = @p2counter);
            INSERT INTO temp2 (Toode_id, Pardiliik1_id, Pardiliik2_id, diff) VALUES
              (@tcounter, @p1counter, @p2counter, ABS(@diff));
            SET @p2counter = @p2counter + 1;
            ITERATE liiged2;
          END IF;
          LEAVE liiged2;
        END LOOP liiged2;
        SET @p1counter = @p1counter + 1;
        ITERATE liiged1;
      END IF;
      LEAVE liiged1;
    END LOOP liiged1;
    SET @tcounter = @tcounter + 1;
    ITERATE tooded;
  END IF;
  LEAVE tooded;
END LOOP tooded;
CREATE TABLE temp3 (Pardiliik1_id int, Pardiliik2_id int, diff float);
SET @p1counter = 1;
SET @p2counter = 1;
pd1: LOOP
  IF (@p1counter <= @maxp) THEN
    SET @p2counter = @p1counter + 1;
    pd2: LOOP
      IF (@p2counter <= @maxp) THEN
        INSERT INTO temp3 (Pardiliik1_id, Pardiliik2_id, diff) VALUES
          (@p1counter, @p2counter, (SELECT sum(t1.diff) FROM
            (SELECT diff from temp2 WHERE Pardiliik1_id = @p1counter AND
              Pardiliik2_id = @p2counter) AS t1));
        SET @p2counter = @p2counter + 1;
        ITERATE pd2;
      END IF;
      LEAVE pd2;
    END LOOP pd2;
    SET @p1counter = @p1counter + 1;
    ITERATE pd1;
  END IF;
  LEAVE pd1;
END LOOP pd1;
(SELECT Pardiliik.Nimetus FROM Pardiliik, temp3
WHERE temp3.Pardiliik2_id = Pardiliik.id
ORDER BY diff DESC LIMIT 1)
UNION
```

```
    (SELECT Pardiliik.Nimetus FROM Pardiliik, temp3
    WHERE temp3.Pardiliik1_id = Pardiliik.id
    ORDER BY diff DESC LIMIT 1);
    DROP TABLE temp;
    DROP TABLE temp2;
    DROP TABLE temp3;
  END; ||

CALL erinevadMaitsed();
```

Väljund:

	Nimetus
1	Branta canadensis
2	Somateria mollissima

5) Triggeri loomine, mis toote kustutamisel kustutab selle toote esinemised poodides.

```
CREATE TRIGGER onToodeDelete
  AFTER delete ON Toode
  FOR EACH ROW
  BEGIN
    DELETE FROM PoesOnToode WHERE Toode_id = old.id;
  END
```

Väljund: väljund puudub