Andmed mitmes tabelis

Vaid muudetava tekstilise sisuga lehestikes võib hakkama saada ühe andmetabeliga. Samuti piisab ühest tabelist juhul, kui hoitakse ja näidatakse ühetüübilisi andmeid. Kui aga tahta veebi kaudu tööle panna vähegi mitmekesisemat infosüsteemi, siis jääb varsti ühest tabelist kitsaks. Mõnedki asjad küll saab veel ära ajada mitme eraldiseisva tabeliga. Ning veidi kavaldades õnnestub ka küllalt mitmekülgseid andmeid ühte tabelisse toppida - kui näiteks ei oska andmeid eri tabelitesse jagada ja neid omavahel siduda. Mingist hetkest aga andmete kokkuväänamine enam ei õnnestu ning siis hakkab tabeleid kergesti ja hulgem juurde tulema. Mugav on ju arvestada, kui ühes tabelis on vaid samatüübilised asjad.

Ühe valitava omadusega nähtuse või eseme kirjeldamisel võib hakkama saada kahe andmetabeliga. Kaht tüüpi objekte (nt. inimesi ja reisikohti) mitut moodi omavahel kombineerides kulub vähemasti kolm tabelit (üks inimeste tarbeks, üks kohtade kirjeldamiseks ning kolmandas tabelis on kirjas reisid, kes millal kus käis). Kasutajate, õiguste, pea- ja alateemadega foorumit kokku pannes kulub neli-viis tabelit. Ning lihtsalt võrdluseks, et kooli õppeainetes läbitavaid teemasid mitmelt poolt otsida võimaldav süsteem kasvas paarikümne andmetabeli suuruseks. Kui juures oli veel palga- ja koormusearvestus ning muu personalihaldus, siis üsna pea kasvas infosüsteemi maht saja andmetabelini.

Tabelite rohkust iseenesest ei maksa karta. Kui nad on arusaadavalt nimetatud ning mõne skeemi peal näha, mis kuidas millega suhtleb, siis on paljude selgete tabelitega süsteem siiski päris mugavalt kasutatav. Enamasti rakenduse juures kasutatakse korraga neist ikka väiksemat, mõne tabeliga alamhulka. Vahel harva tekivad päringud, kus korraga on andmeid vaja kümnekonnast tabelist.

Kaubad ja kaubagrupid

Siin alustame taas võimalikult lihtsalt - seome omavahel kokku kaks andmetabelit. Ühes on kirjas olemasolevad kaubagrupid - igal grupil id ja grupinimi. Teises tabelis on tooted, kusjuures iga toode kuulub kindlasse gruppi. Selliselt kirja pannes on võimalik andmeid viisakasti vaadata gruppide kaupa. Pole karta, et üks kirjutab tööriistad väikese, teine suure tähega ning arvuti jaoks võivad vastavates gruppides olevad asjad sootuks erinevatena tunduda. Kui ikka grupi nimi valitakse toote juurde rippmenüüst, siis pole võimalik selle valimise juures enam grupi nime vigaselt kirjutada. Näitrakenduse teemaks tuleb

```
Kaupade otsing poes (tootekataloog)
```

Rakenduse jaoks vajame kahte andmetabelit.

```
kaubad:
id nimetus kaubagrupi_id hind

kaubagrupid:
id grupinimi
```

Sisuandmed näiteks:

```
kaubagrupid

1 tellised
2 katusematerjal
```

```
3 vineer

kaubad

1 ahjutellis      1    8.20
2 fassaaditellis  1    7.50
3 bituumenrull    2  520.00
```

Kaupade tabeli kaubagrupi_id näitab kaubagruppide tabeli id-le. Siit saab siis välja lugda, et ahjutellis ja fassaaditellis kuluvad mõlemad gruppi "tellised" ehk gruppi nr 1. Bituumenrull aga läheb teise grupi ehk katusematerjali alla.

Teeme esimese tabeli SQL-lause abil valmis

```
CREATE TABLE kaubagrupid(
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  grupinimi VARCHAR(255)
);
```

ning lisame mõned andmed sisse.

```
INSERT INTO kaubagrupid(grupinimi) VALUES ('tellised');
INSERT INTO kaubagrupid(grupinimi) VALUES ('katusematerjal');
```

Päringuga kontrollime, et andmed ikka kohale jõudsid. Samuti saame teada tekkinud gruppide id-d.

```
mysql> SELECT * FROM kaubagrupid;
+----+----------------+
| id | grupinimi      |
+----+----------------+
|  1 | tellised       |
|  2 | katusematerjal |
+----+----------------+
2 rows in set (0.00 sec)
```

Kui grupid olemas, on põhjust luua kaupade tabel.

```
CREATE TABLE kaubad(
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  nimetus VARCHAR(255),
  kaubagrupi_id INT,
  hind DECIMAL(10, 2)
);
```

Iseenesest me teame, et kaupade tabeli kaubagrupi_id näitab kaubagruppide tabeli id peale. Seda saaks ka SQL lauses märkida (FOREIGN KEY(kaubagrupi_id) REFERENCES kaubagrupid(id)). Kuna aga MySQL vaikimisi juhul nagunii viiteterviklust ei kontrolli, siis pigem on tähtsam, et oma peas meeles oleks, mis millega ja kuidas seotud.

Andmete lisamiseks SQLi kaudu paneme siis praegu lihtsalt vastavad grupinumbrid lausesse kirja.

```
INSERT INTO kaubad (nimetus, kaubagrupi_id, hind) VALUES ('ahjutellis', 1,
8.20);
INSERT INTO kaubad (nimetus, kaubagrupi_id, hind) VALUES ('fassaaditellis', 1,
7.50);
INSERT INTO kaubad (nimetus, kaubagrupi_id, hind) VALUES ('bituumenrull', 2,
520);
```

SELECTi abil kontrollime järgi, et soovitud andmed ikka baasi kohale jõudsid.

```
mysql> SELECT * FROM kaubad;
+----+---------------+---------------+--------+
| id | nimetus       | kaubagrupi_id | hind   |
+----+---------------+---------------+--------+
|  1 | ahjutellis    |             1 |   8.20 |
|  2 | fassaaditellis |             1 |   7.50 |
|  3 | bituumenrull  |             2 | 520.00 |
+----+---------------+---------------+--------+
```

ID-numbrid on küll programmeerija jaoks toredad, kuid tavainimene armastab andmeid siiski sõnalisel kujul lugeda. Järgmise lause abil saab näha kauba nimetust, sõnalist grupinime (mis võetud kaubagruppide tabelist) ning kauba hinda. Selleks kirjutan soovitud tulbad SELECTi järele. FROM-ossa märgin kasutatavad andmetabelid ning WHERE tingimuses panen kirja, kuidas tabelid omavahel seotud on. Siinsel juhul siis tabeli kaubad tulp kaubagrupi_id näitab tabeli kaubagrupid tulbale id. Võimalikest kummagi tabeli ridade kombinatsioonidest näidatakse välja siis vaid need, kus kaupade tabeli kaubagrupi_id kattub kaubagruppide tabeli id-ga.

```
SELECT nimetus, grupinimi, hind
FROM kaubad, kaubagrupid
WHERE kaubad.kaubagrupi_id=kaubagrupid.id;
```

Mugavuse mõttes on hea SQL-lause enne mõnes tekstiredaktoris valmis kirjutada ning alles siis MySQLi viiba otsa kopeerida. Sellisel juhul on mõne vea puhul kergesti võimalik lauset täiendada/parandada ning uuesti katsetada.

```
mysql> SELECT nimetus, grupinimi, hind
    -> FROM kaubad, kaubagrupid
    -> WHERE kaubad.kaubagrupi_id=kaubagrupid.id;
+---------------+----------------+--------+
| nimetus       | grupinimi      | hind   |
+---------------+----------------+--------+
| ahjutellis    | tellised       |   8.20 |
| fassaaditellis | tellised       |   7.50 |
| bituumenrull  | katusematerjal | 520.00 |
+---------------+----------------+--------+
```

Päringu tulemus paistis täiesti ootuspärane olema: tellised kuuluvad telliste gruppi ning bituumenrull katusele.

Mõned SQL-laused veel. Tingimuste abil saab osa ridu peita, ehk siis soovitud välja näidata. Järgnevalt kaubad, mille tükihind on alla kümne krooni.

```
mysql> SELECT nimetus, hind FROM kaubad WHERE hind<10.00;
+---------------+------+
| nimetus       | hind |
+---------------+------+
| ahjutellis    | 8.20 |
| fassaaditellis | 7.50 |
+---------------+------+
```

Tekstide puhul on tänuväärne funktsioon nimega LIKE. Päringusõnas tähendab protsent suvalist hulka suvalisi sümboleid. Ehk siis praegu otsitakse kõik kaubad, mille nimes sisaldub sõna tellis.

```
mysql> SELECT nimetus, hind FROM kaubad WHERE nimetus LIKE '%tellis%';
+---------------+------+
| nimetus       | hind |
+---------------+------+
| ahjutellis    | 8.20 |
| fassaaditellis | 7.50 |
```

```
+----------------+------+
```

Mitme tabeli ühendamine ning piirang päringu juures võivad kehtida ka üheaegselt.

```
SELECT nimetus, grupinimi, hind
 FROM kaubad, kaubagrupid
 WHERE kaubad.kaubagrupi_id=kaubagrupid.id
 AND nimetus LIKE '%tellis%';

+----------------+-----------+------+
| nimetus        | grupinimi | hind |
+----------------+-----------+------+
| ahjutellis     | tellised  | 8.20 |
| fassaaditellis | tellised  | 7.50 |
+----------------+-----------+------+
```

Kui päringusse lisada ka id-tulp, siis juhtub tulema veateade:

```
Column 'id' in field list is ambiguous
```

Kuna mõlemal tabelil on vastava nimega tulp olemas, et teata, kas soovitakse näha kauba või kaubagrupi id-numbrit. Selle vastu aitab, kui tulba ette kirjutada tabeli nimi ehk siis praegusel juhul kaubad.id.

Ülesandeid

* Tee näide läbi/võta eeskujuks
* Koosta tabel maakondade jaoks (id, maakonnanimi, maakonnakeskus)
* Koosta tabel inimeste andmete jaoks, maakonnale viidatakse võõrvõtme abil (id, eesnimi, perekonnanimi, maakonna_id)
* Näita tabelis olevate inimeste andmed koos nende paiknemise maakonna nimedega ekraanile.
* Näita välja vaid A-ga algava eesnimega inimesed.
* Näita ekraanile vaid inimesed, kelle maakonna keskus on Haapsalu

Seotud tabelite andmed veebilehel

Siin näites püüame samuti hoida kujunduse ja koodi võimalikult lahus. Koodiosa paigutame faili nimega

```
abifunktsioonid.php
```

Eraldi seletust vajab ehk andmete väljastus funktsioonist. Varasemas näites tagastati funktsioonist vaid teate sisu - selleks piisas return-käsust. Siin tuleb aga iga kauba kohta neli väärtust: id, nimetus, grupinimi ja hind. Lisaks püüame funktsioonist korraga kätte saada kõikide kaupade andmed, et neid siis veebilehel mugavasti sobival kujul kuvada saaks.

Mitme samatüübilise väärtuse hoidmiseks aitab massiiv - nii ka siin. Enne andmete läbikäimise tsüklit tehakse uus massiiv nimega $hoidla.

```
    $hoidla=array();
```

Kauba andmete ühe kesta sisse kokku panekuks sobib stdClass-tüüpi objekt. Selle saab new-käsuga luua ning hiljem sinna ükshaaval omadusi külge panna. Olemasolevatest muutujatest võtame andmed ning paneme need kaubaisendi väljadeks. Tekstilisi andmeid töötleme käsuga htmlspecialchars, et teksti sees olevad erisümbolid ei pääseks veebilehe sisu segama.

```
      $kaup=new stdClass();
      $kaup->id=$id;
      $kaup->nimetus=htmlspecialchars($nimetus);
      $kaup->grupinimi=htmlspecialchars($grupinimi);
      $kaup->hind=$hind;
```

Iga kauba puhul kui selle andmed käes, lisatakse kaubaobjekt massiivi lõppu.

```
      array_push($hoidla, $kaup);
```

Funktsioonist väljudes tagastatakse hoidla koos kaupade andmetega.

```
    return $hoidla;
```

Funktsiooni töö kontrollimiseks on algul hea ta käivitada. Selleks abifunktsioonide faili lõpus vastav lõik. Märgend <pre> määrab, et veebilehel väljastades oleksid kõik tähed (ka tühikud) kindla laiusega (preformatted text). Käsklus print_r (rekursiivne print) trükib etteantud objekti või massiivi andmed sügavuti välja, ehk siis näitab kõik, mis sealt seest saada on.

```
<pre>
<?php
  print_r(kysiKaupadeAndmed());
?>
</pre>
```

Nüüd siis abifunktsioonid.php tervikuna.

```
<?php
  $yhendus=new mysqli("localhost", "jaagup", "xxxxxx", "jaagup");

  function kysiKaupadeAndmed(){
    global $yhendus;
    $kask=$yhendus->prepare("SELECT kaubad.id, nimetus, grupinimi, hind
       FROM kaubad, kaubagrupid
       WHERE kaubad.kaubagrupi_id=kaubagrupid.id");
    //echo $yhendus->error;
    $kask->bind_result($id, $nimetus, $grupinimi, $hind);
    $kask->execute();
    $hoidla=array();
    while($kask->fetch()){
      $kaup=new stdClass();
      $kaup->id=$id;
      $kaup->nimetus=htmlspecialchars($nimetus);
      $kaup->grupinimi=htmlspecialchars($grupinimi);
      $kaup->hind=$hind;
      array_push($hoidla, $kaup);
    }
    return $hoidla;
  }
?>
<pre>
<?php
  print_r(kysiKaupadeAndmed());
```

```
?>
</pre>
```

Faili väljund parasjagu baasis olevate andmete põhjal. Käsu print_r tulemus näitab, et väljatrükitud objekt oli massiiv, millel indeksid 0, 1 ja 2. Ning mille igaks elemendiks on objekt klassist stdClass väljadega id, nimetus, grupinimi ja hind parasjagu sissepandud väärtustega.

```
Array
(
    [0] => stdClass Object
        (
            [id] => 1
            [nimetus] => ahjutellis
            [grupinimi] => tellised
            [hind] => 8.20
        )

    [1] => stdClass Object
        (
            [id] => 2
            [nimetus] => fassaaditellis
            [grupinimi] => tellised
            [hind] => 7.50
        )

    [2] => stdClass Object
        (
            [id] => 3
            [nimetus] => bituumenrull
            [grupinimi] => katusematerjal
            [hind] => 520.00
        )

)
```

Kui funktsiooni töö kontrollitud ja õigeks loetud, siis on hea pre-ga tähistatud osa abifunktsioonide failist välja võtta, et see muude failide poolt välja kutsudes segama ei hakkaks.

Järgnevalt kasutame failis kaubaotsing.php loodud abifunktsiooni teenuseid ning näitame tabelis olevad kaubad ekraanile. Kõigepealt tuleb kasutamiseks lugeda require-käsuga abifunktsioonid käivituvasse faili (kaubaotsing.php) sisse. Ning et kaubad oleksid meil lehe väljatrüki juures pidevalt käepärast, selleks küsime funktsiooni poolt väljastatud andmekogumi muutujasse nimega $kaubad.

```
<?php
  require("abifunktsioonid.php");
  $kaubad=kysiKaupadeAndmed();
?>
```

Hiljem andmeid välja trükkides saab siis kaupade massiivi elemendid ükshaaval ette võtta ning iga kauba andmed sobivasse lahtrisse paigutada. Kuna eelnevas funktsioonis on juba kauba nimetusele ja grupinimele pandud ümber htmlspecialchars'i tehtud muudatused, siis võib need väärtused siin rahus otse välja trükkida.

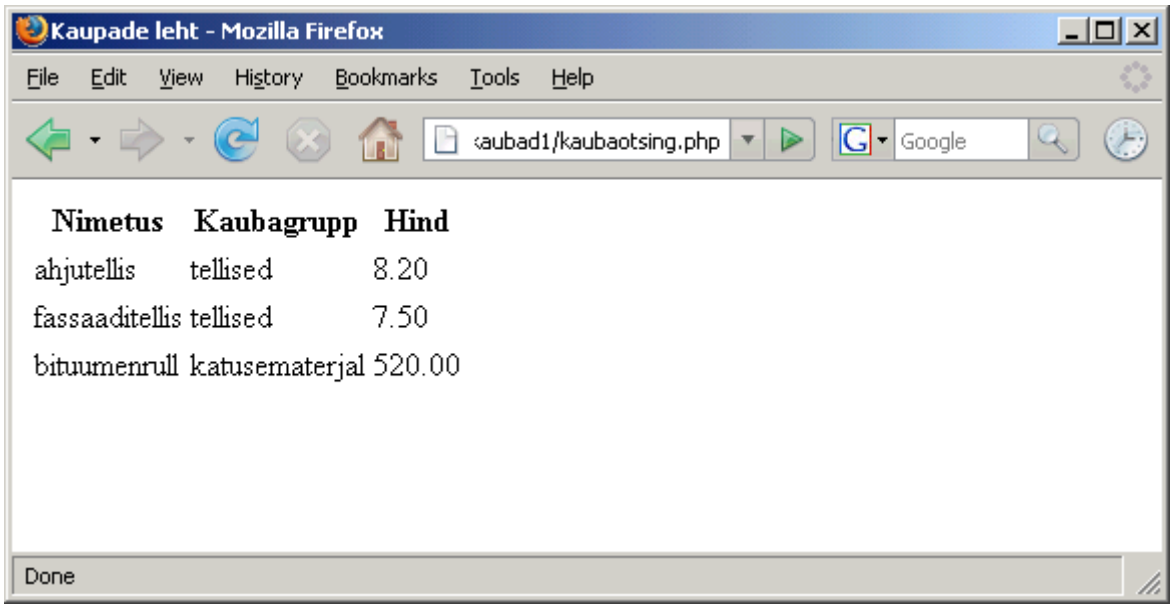
```
        <?php foreach($kaubad as $kaup): ?>
          <tr>
            <td><?=$kaup->nimetus ?></td>
            <td><?=$kaup->grupinimi ?></td>
```

```
          <td><?=$kaup->hind ?></td>
        </tr>
      <?php endforeach; ?>
```

Ning kaubaotsingu fail tervikuna

```
<?php
  require("abifunktsioonid.php");
  $kaubad=kysiKaupadeAndmed();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
      <title>Kaupade leht</title>
      <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  </head>
  <body>
     <table>
       <tr>
         <th>Nimetus</th>
         <th>Kaubagrupp</th>
         <th>Hind</th>
       </tr>
       <?php foreach($kaubad as $kaup): ?>
         <tr>
           <td><?=$kaup->nimetus ?></td>
           <td><?=$kaup->grupinimi ?></td>
           <td><?=$kaup->hind ?></td>
         </tr>
       <?php endforeach; ?>
     </table>
  </body>
</html>
```

Ja tema töö tulemus arvutiekraanil.



Abifunktsioone on täiendamise korral vaja sageli testida. Pidevalt pre-osa sisse ja välja tõsta on tülikas. Selle asemel võib tingimuslausega kontrollida, et kas abifunktsioonid.php käivitatakse iseseisvalt või mõne teise faili kaudu. Esimesel juhul näidatakse testandmed ekraanile, muidu mitte. Tegeliku veebilehitseja aadressirealt käivitatava faili nime leiab muutujast `$_SERVER["PHP_SELF"]` -

koos kogu pika URLiga. Järgnevas käsus tükeldatakse see URL explode abil kaldkriipsu kohtadelt massiivielementideks ning `array_pop` käsu abil võetakse viimane element ehk failinimi ise. Tingimusega kontrollitakse, et kui aadressirealt käivitatava faili nimi on abifunktsioonid.php, siis pannakse pre-märgendi sees olev testosa käima, muidu mitte.

```
  if( array_pop(explode("/", $_SERVER["PHP_SELF"]))=="abifunktsioonid.php"):
?>
<pre>
<?php
  print_r(kysiKaupadeAndmed());
?>
</pre>
<?php endif ?>
```

Nõnda siis abifunktsioonid.php uuel kujul, kus faili lõpus testosa käivitatakse ainult faili otse veebilehitsejas vaadates. Teise faili kaudu funktsioon sisse lugedes aga muud lehte segavat lõiku ei teki.

```
<?php
  $yhendus=new mysqli("localhost", "jaagup", "xxxxxx", "jaagup");

  function kysiKaupadeAndmed(){
    global $yhendus;
    $kask=$yhendus->prepare("SELECT kaubad.id, nimetus, grupinimi, hind
       FROM kaubad, kaubagrupid
       WHERE kaubad.kaubagrupi_id=kaubagrupid.id");
    //echo $yhendus->error;
    $kask->bind_result($id, $nimetus, $grupinimi, $hind);
    $kask->execute();
    $hoidla=array();
    while($kask->fetch()){
      $kaup=new stdClass();
      $kaup->id=$id;
      $kaup->nimetus=htmlspecialchars($nimetus);
      $kaup->grupinimi=htmlspecialchars($grupinimi);
      $kaup->hind=$hind;
      array_push($hoidla, $kaup);
    }
    return $hoidla;
  }

  //---------------
  if( array_pop(explode("/", $_SERVER["PHP_SELF"]))=="abifunktsioonid.php"):
?>
<pre>
<?php
  print_r(kysiKaupadeAndmed());
?>
</pre>
<?php endif ?>
```

Ülesandeid

* Tee/mõtle näide läbi.
* Loo/otsi tabel maakonnad (id, maakonnanimi, maakonnakeskus) ning inimesed (id, eesnimi, perekonnanimi, maakonna_id)
* Näita veebilehele inimeste andmed koos nende paiknemise maakonna nimega.

* Andmebaasipäring paiguta eraldi failis olevasse funktsiooni nagu eelnevas näites.

Sortimine

Suuremast andmehulgast sobiva leidmisel aitab sageli andmete sortimine soovitud tunnuse järgi. Telefoniraamatust näiteks nime järgi numbri leidmine on suhteliselt lihtne toimetus. Numbri järgi kindlaks tegemine, et kellele see kuulub, nõuab aga palju rohkem vaeva. Kõik lihtsalt sellepärast, et telefoniraamat on sorditud nimede järgi.

Järgnevas näites täiendame kaupade loetelu otsinguvõimalusega. Andmebaasipõhises lehestikus ühe tulba järgi sortida on lihtne - tuleb vaid SELECT-lause lõppu kirjutada ORDER BY tulbanimi ning andmed saabuvadki sobivas järjestuses. Siin aga pakume kasutajale võimalust sortida andmed nimetuse, grupinime või hinna järgi - vajutades vastava tulba pealkirjale. Sellisel juhul tuleb vastavalt kasutaja valikule saada SQL-lausesse sobiva tulba nimi.

Koodi täiendades on mugav, kui õnnestub teha nii, et olemasolevad funktsioonid ka vanades kasutuskohtades tööle jäävad. Vanas variandis ei andnud me funktsioonile kysiKaupadeAndmed ette midagi - väljastati lihtsalt baasis leiduvad kaubad. Nüüd aga soovime, et funktsioon väljastaks kaubad sordituna etteantud tulba järgi. Funktsioonile saab andmeid ette anda parameetrite kaudu. Kui lihtsalt lisada funktsioonile kohustuslik parameeter sortimistulba määramiseks, siis ilma selle parameetrita ei saaks funktsiooni enam välja kutsuda. Siin näites pole sellest suurt muret, sest täiendus tuleks viia sisse vaid ühte kohta - sinna kus kaubaotsing.php failis andmeid teise faili järgi küsitakse. Mõnes suuremas rakenduses võidakse sama funktsiooni kasutada kümnetes ja sadades kohtades. Ning vahel kasutatakse abifunktsioonide teeke kogumikuna mõne teise rakenduse juures, millest abifunktsioonide loojal ei pruugi aimugi olla. Sellisel juhul tekitaks funktsioonile parameetrite lisamine paljudes kohtades. Mõnikord pole sellest pääsu - eriti kui jõudluse või turvalisuse huvides funktsioonilt osa oskusi maha võetakse. Parameetrite lisamisel aga saab üldjuhul muudele koodifailidele murede tekitamisest hoiduda pannes loodud parameetrile vaikimisi väärtuse. Siin näiteks siis lisatakse parameeter $sorttulp. Ning juhul, kui seda parameetrit funktsioonile ette ei anta (nt. siiamaani loodud kasutavate failide korral), sellisel juhul antakse sortimistulbale vaikimisi väärtuseks "nimetus", st. et väljastatavad kaubad sorditakse nimetuse järgi tähestikulisse järjekorda.

```
function kysiKaupadeAndmed($sorttulp="nimetus")
```

Veebist saabuvaid andmeid ei või kunagi usaldada - näpuvea või pahatahtlikkuse tõttu võib sealt saabuvates päringutes sisalduda tont-teab-mida. Et saabuvad andmed me rakendust rikkuda ei saaks, tasub ära märkida, milliste tulpade järgi on lubatud sortida - praegu siis pikkus, grupinimi ja hind.

```
    $lubatudtulbad=array("nimetus", "grupinimi", "hind");
```

Kui juhtub, et palutakse andmeid järjestada mõne muu (seni olematu) tunnuse järgi, siis lihtsalt katkestatakse funktsiooni töö veateatega. Kas ja mida väljakutsuv programm selle veateatega peale hakkab, on juba tema mure. Vähemasti ei satu sobimatud andmed edasi me SQL-lausessese andmebaasi sisse pahandust tegema.

```
    if(!in_array($sorttulp, $lubatudtulbad)){
      return "lubamatu tulp";
    }
```

SELECT-lause on lihtne ja suhteliselt sarnane eelmise näite variandiga. Lihtsalt lõppu on tulnud määrang "ORDER BY $sorttulp", mis siis vastavalt päringule asendatakse kas nimetuse, grupinime

või hinnaga.

```
    $kask=$yhendus->prepare("SELECT kaubad.id, nimetus, grupinimi, hind
       FROM kaubad, kaubagrupid
       WHERE kaubad.kaubagrupi_id=kaubagrupid.id
       ORDER BY $sorttulp");
```

Väljakommenteeritud rida vea kuvamiseks on mugav vahend jälgimaks, kui miski ei taha tööle hakata. Selle käsu kaudu saab vajadusel välja trükkida MySQLi poolt avastatud vigade selgitused. Töötavasse lõppversiooni pole koodi soovitatav neid käsklusi sisse jätta - serveripoolsed veateated võivad hõlbustada häkkerite tööl. Enese jaoks arenduse käigus vigade kohta aimu saamiseks on selliste veateadete nägemine aga igati omal kohal.

```
    //echo $yhendus->error;
```

Ülejäänud koodiosa jäi samaks - kergema kasutamise ja kopeerimise huvides on ta aga ka siia pandud.

```
<?php
  $yhendus=new mysqli("localhost", "jaagup", "xxxxxx", "jaagup");

  function kysiKaupadeAndmed($sorttulp="nimetus"){
    global $yhendus;
    $lubatudtulbad=array("nimetus", "grupinimi", "hind");
    if(!in_array($sorttulp, $lubatudtulbad)){
      return "lubamatu tulp";
    }
    $kask=$yhendus->prepare("SELECT kaubad.id, nimetus, grupinimi, hind
       FROM kaubad, kaubagrupid
       WHERE kaubad.kaubagrupi_id=kaubagrupid.id
       ORDER BY $sorttulp");
    //echo $yhendus->error;
    $kask->bind_result($id, $nimetus, $grupinimi, $hind);
    $kask->execute();
    $hoidla=array();
    while($kask->fetch()){
      $kaup=new stdClass();
      $kaup->id=$id;
      $kaup->nimetus=htmlspecialchars($nimetus);
      $kaup->grupinimi=htmlspecialchars($grupinimi);
      $kaup->hind=$hind;
      array_push($hoidla, $kaup);
    }
    return $hoidla;
  }

  //---------------
  if( array_pop(explode("/", $_SERVER["PHP_SELF"]))=="abifunktsioonid.php"):
?>
<pre>
<?php
  print_r(kysiKaupadeAndmed("hind"));
?>
</pre>
<?php endif ?>
```

Loodud abifunktsioone kasutame endiselt andmeotsingu lehel. Lehe algusesse tuleb juurde kontroll, kas aadressireal olevaks sort-parameetriks väärtuseks on saabunud midagi. Kui jah, siis palutakse andmed sordituna selle tunnuse järgi, muul juhul palutakse lihtsalt andmeid, kusjuures praegu siis sorditakse need funktsiooni vaikimisi parameetri ehk nimetuse järgi.

```
  if(isSet($_REQUEST["sort"])){
     $kaubad=kysiKaupadeAndmed($_REQUEST["sort"]);
  } else {
     $kaubad=kysiKaupadeAndmed();
```

```
  }
```

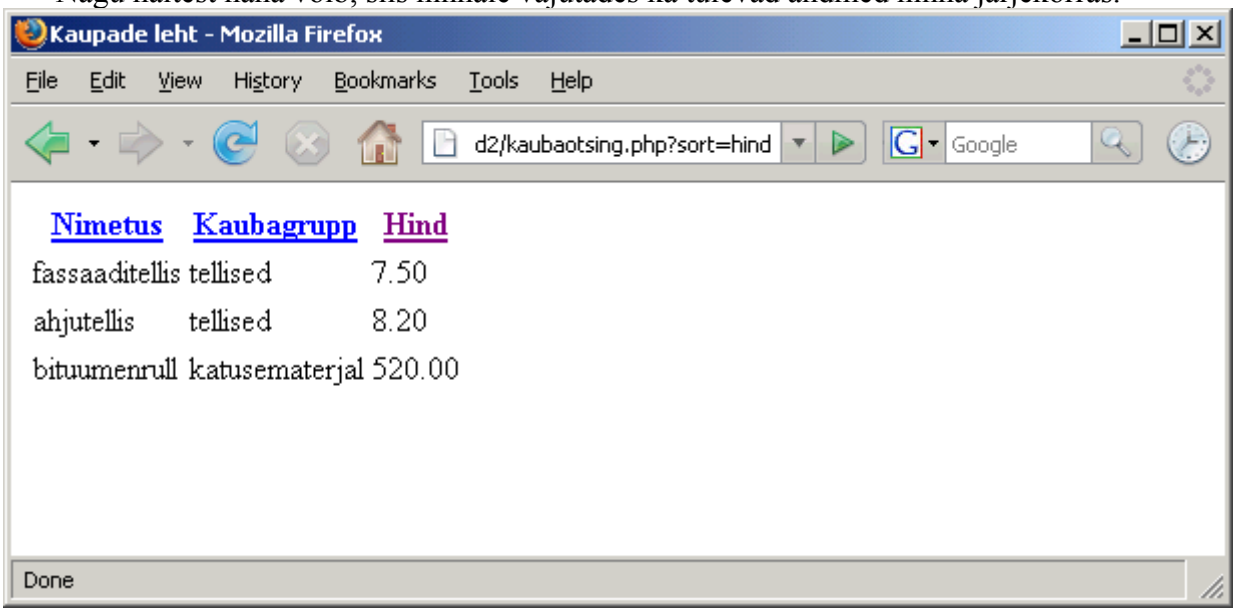
Sortimistulba mugavaks valimiseks tuleb tulbad vastavateks viideteks teha, igaüks saadab kaubaotsing.php lehele parameetri sort sobiva väärtusega.

```
        <tr>
          <th><a href="kaubaotsing.php?sort=nimetus">Nimetus</a></th>
          <th><a href="kaubaotsing.php?sort=grupinimi">Kaubagrupp</a></th>
          <th><a href="kaubaotsing.php?sort=hind">Hind</a></th>
        </tr>
```

Muu fail ikka eelmise näitega sarnane.

```
<?php
  require("abifunktsioonid.php");
  if(isSet($_REQUEST["sort"])){
     $kaubad=kysiKaupadeAndmed($_REQUEST["sort"]);
  } else {
     $kaubad=kysiKaupadeAndmed();
  }
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
      <title>Kaupade leht</title>
      <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  </head>
  <body>
     <table>
       <tr>
         <th><a href="kaubaotsing.php?sort=nimetus">Nimetus</a></th>
         <th><a href="kaubaotsing.php?sort=grupinimi">Kaubagrupp</a></th>
         <th><a href="kaubaotsing.php?sort=hind">Hind</a></th>
       </tr>
       <?php foreach($kaubad as $kaup): ?>
         <tr>
           <td><?=$kaup->nimetus ?></td>
           <td><?=$kaup->grupinimi ?></td>
           <td><?=$kaup->hind ?></td>
         </tr>
       <?php endforeach; ?>
     </table>
  </body>
</html>
```

Nagu näitest näha võib, siis hinnale vajutades ka tulevad andmed hinna järjekorras.



Ülesandeid

* Tee näide läbi
* Pane sarnane sortimisvõimalus tööle eelneva näite juures tehtud inimeste ja nende paiknemismaakondade rakenduse juures.

Otsimine

Sortimine aitab andmete juures, kus tulba tekst algab otsitava väärtusega. Kui aga otsitav lõik võib asuda teksti sees, siis paljas tekstide või arvude tähestikulisse või kasvavasse järjekorda seadmine ei aita soovitud rida leida. Sellisel puhul on abiliseks otsimisvõimalus. Tänapäevastel veebilehestikel on see paljudel sisse ehitatud. Ning kui ka pole, siis mõnevõrra saab toetuda ka Google otsingule, kirjutades otsingu ette site:masinanimi. Näiteks otsing "site:minitorn.tlu.ee php" annab minitorn.tlu.ee masinas PHPga seotud lehed - niipalju, kui otsimootor neid sealt leidnud on.

Oma särk aga ikka ihule kõige lähemal ning isetehtud otsingu peale võib ka ehk kindlam olla, et ta just vajalikest kohtadest otsib. Funktsioonile tuleb siis juurde järjekordne parameeter. Ning kui otsisõna ei määrata, siis selle väärtuseks saab tühi tekst - ehk osa, mis igas tekstis leidub ning välja kuvatakse kõik andmed.

```
  function kysiKaupadeAndmed($sorttulp="nimetus", $otsisona=""){
```

PHP murelapseks on üle veebi saabuvate sümbolite varjestamine langjoontega. Toimetus oli vajalik saabuvate andmete paigutamiseks MySQLi päringusse. MySQLi seda aga üldjuhul ei vaja ning serveri konfiguratsioonis võib olla erisümbolite varjestus maha võetud. Et küsimärgiga parameeter mugavalt tekstisobivust otsivasse LIKE-lausesse ei lähe, tuleb siin otsitav parameeter otse SQLi paigutada. Kui ei tea, kas varjestavad langjooned on ees või mitte, siis üheks võimaluseks on need maha võtta stripslashes-käsuga ning hiljem uuesti addslashes-käsuga tagasi panna. Puuduva varjestuse puhul pole ka stripslashes-il midagi maha võtta, teine peale paneb ikka, et oleks andmebaasilauses kõik korrektne.

```
    $otsisona=addslashes(stripslashes($otsisona));
```

Otsinguosa saab lause WHERE-ossa olemasolevale tingimusele otsa liita. Siin otsime korraga nimetuse ja grupinime seest - ükskõik kummas otsitav lõik leitakse, sobiv rida kuvatakse ikka tulemusena. Et kaupade tabeli kaubagrupi_id järgi kaubagrupi tabeli id-le viitamist ikka alati kontrollitaks, selleks peab ORidega kahe tulba järgi otsing eraldi sulgudes olema.

```
    $kask=$yhendus->prepare("SELECT kaubad.id, nimetus, grupinimi, hind
       FROM kaubad, kaubagrupid
       WHERE kaubad.kaubagrupi_id=kaubagrupid.id
        AND (nimetus LIKE '%$otsisona%' OR grupinimi LIKE '%$otsisona%')
       ORDER BY $sorttulp");
```

Ning fail tervikuna.

```
<?php
  $yhendus=new mysqli("localhost", "jaagup", "xxxxxx", "jaagup");

  function kysiKaupadeAndmed($sorttulp="nimetus", $otsisona=""){
    global $yhendus;
    $lubatudtulbad=array("nimetus", "grupinimi", "hind");
    if(!in_array($sorttulp, $lubatudtulbad)){
      return "lubamatu tulp";
```

```
    }
    $otsisona=addslashes(stripslashes($otsisona));
    $kask=$yhendus->prepare("SELECT kaubad.id, nimetus, grupinimi, hind
       FROM kaubad, kaubagrupid
       WHERE kaubad.kaubagrupi_id=kaubagrupid.id
        AND (nimetus LIKE '%$otsisona%' OR grupinimi LIKE '%$otsisona%')
       ORDER BY $sorttulp");
    //echo $yhendus->error;
    $kask->bind_result($id, $nimetus, $grupinimi, $hind);
    $kask->execute();
    $hoidla=array();
    while($kask->fetch()){
      $kaup=new stdClass();
      $kaup->id=$id;
      $kaup->nimetus=htmlspecialchars($nimetus);
      $kaup->grupinimi=htmlspecialchars($grupinimi);
      $kaup->hind=$hind;
      array_push($hoidla, $kaup);
    }
    return $hoidla;
  }

  //---------------
  if( array_pop(explode("/", $_SERVER["PHP_SELF"]))=="abifunktsioonid.php"):
?>
<pre>
<?php
  print_r(kysiKaupadeAndmed("hind", "fass\\aad"));
?>
</pre>
<?php endif ?>
```

Kaubaotsingu failis tuleb juurde koht otsinguteksti sisestamiseks. Ning sisestuselemendil peab ümber olema vorm koos action'iga määramaks, kuhu faili sisestatud andmed saata.

```
   <form action="kaubaotsing.php">
     Otsi: <input type="text" name="otsisona" />
         ...
   </form>
```

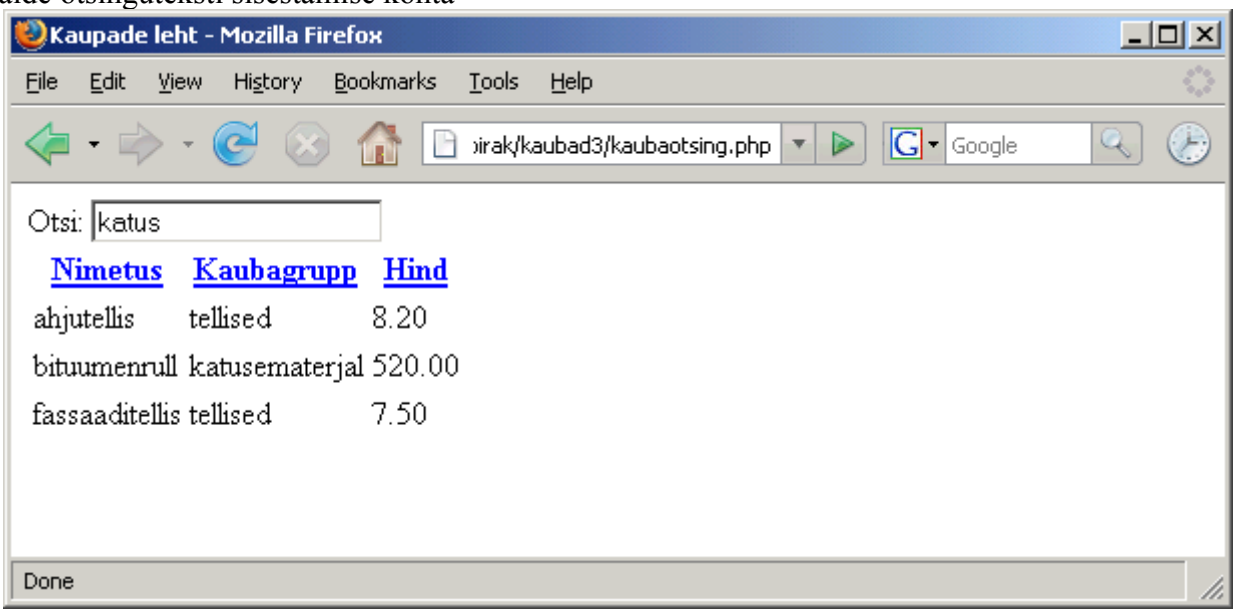
Kuna veebilehe puhul ei ole nüüd enam teada, kas üldse ja kumb parameeter on väärtustatud, siis on heaks mooduseks neile ka PHP-lehel vaikeväärtused anda. Hädapärast võiks töötada ka vorm

`$kaubad=kysiKaupadeAndmed($_REQUEST["sort"], $_REQUEST["otsisona"])`, aga kui vastavad parameetrid lehel puudu (nt. esimest korda avades), siis võib PHP server vastava konfiguratsiooni puhul hakata hoiatusi andma, et küsitakse olematuid väärtusi massiivist. Nõnda siis järgmine lehekuju koos algusega ikka kindlam.

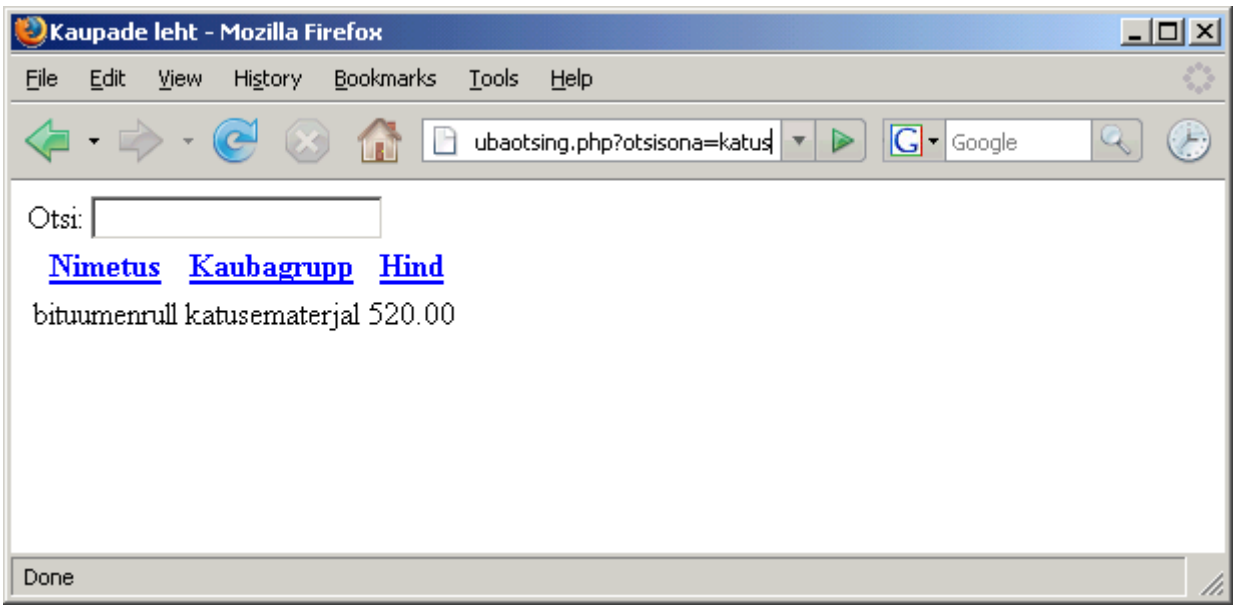
```
<?php
  require("abifunktsioonid.php");
  $sorttulp="nimetus";
  $otsisona="";
  if(isSet($_REQUEST["sort"])){
     $sorttulp=$_REQUEST["sort"];
  }
  if(isSet($_REQUEST["otsisona"])){
     $otsisona=$_REQUEST["otsisona"];
  }
  $kaubad=kysiKaupadeAndmed($sorttulp, $otsisona);
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
      <title>Kaupade leht</title>
```

```
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  </head>
  <body>
   <form action="kaubaotsing.php">
     Otsi: <input type="text" name="otsisona" />
     <table>
       <tr>
         <th><a href="kaubaotsing.php?sort=nimetus">Nimetus</a></th>
         <th><a href="kaubaotsing.php?sort=grupinimi">Kaubagrupp</a></th>
         <th><a href="kaubaotsing.php?sort=hind">Hind</a></th>
       </tr>
       <?php foreach($kaubad as $kaup): ?>
         <tr>
           <td><?=$kaup->nimetus ?></td>
           <td><?=$kaup->grupinimi ?></td>
           <td><?=$kaup->hind ?></td>
         </tr>
       <?php endforeach; ?>
     </table>
   </form>
  </body>
</html>
```

Näide otsinguteksti sisestamise kohta



Ning aadressiriba peale jõudnud sõna järgi otsingutulemus.



Ülesandeid

* Tee näide läbi
* Lisa eelnevale inimeste andmete rakendusele otsing teksti järgi

Haldamine

Seni sortisime ja otsisime neid andmeid, mis juba andmetabelis olemas. Kui andmete sisestamine käib sisevõrgus oleva süsteemi kaudu, siis võibki veebiliides nõnda lihtsa osaga piirduda. Kui aga veebi kaudu tuleb ka andmeid lisada ja muuta, siis vaja mõnevõrra rohkem koodi kirjutada.

Kuna kaubagrupid on määratud eraldi tabelis ning võimaluse korral paigutatakse kaup juba olemasolevasse gruppi, siis paremaks mooduseks on kaubagrupi nime mitte tekstina sisse kirjutada, vaid olemasolevate hulgast rippmenüüst valida. Ning kui sobiv grupp puudub, saab selle pigem kusagilt eraldi kohast juurde panna - nii et seda edaspidi kõikide lisatavate kaupade juures kasutada saab.

Kood sellise lisamisvormi loomiseks võiks välja näha nagu allpool. Nimetus ja nind tulevad tekstiväljadest. Tasub märkida, et nii kauba lisamise nupule kui ka grupi lisamise nupule on pandud nimi. Sellisel puhul saab serverisse saadetud andmete põhjalt vaadata, et millisele nupule vajutati. Vormi action-parameetris määratud veebilehele saadetakse kaasa nupu nimega parameeter, mille väärtuseks on nupu peale kirjutatud tekst.

Rippmenüü loomiseks on tehtud eraldi alamprogramm. Kuna tegemist suhteliselt levinud ja korduva toiminguga, siis on otstarbekas SQL-lause põhjal rippmenüü loomine eraldi alamprogrammi kirjutada ning seda hiljem vajalikes kohtades kasutada. Üks põhjus, et nõnda vähem koodi kirjutada. Teine põhjus, et kui nõnda saab menüü loomise korra viisakalt valmis tehtud, siis järgmistel kordadel pole enam vaja karta, et äkki sinna mõni näpuviga kergesti sisse satub.

```
<form action="kaubahaldus.php">
  <h2>Kauba lisamine</h2>
  <dl>
    <dt>Nimetus:</dt>
    <dd><input type="text" name="nimetus" /></dd>
    <dt>Kaubagrupp:</dt>
    <dd><?php
      echo looRippMenyy("SELECT id, grupinimi FROM kaubagrupid",
                        "kaubagrupi_id");
    ?>
    </dd>
    <dt>Hind:</dt>
    <dd><input type="text" name="hind" /></dd>
  </dl>
  <input type="submit" name="kaubalisamine" value="Lisa kaup" />
  <h2>Grupi lisamine</h2>
  <input type="text" name="uuegrupinimi" />
  <input type="submit" name="grupilisamine" value="Lisa grupp" />
</form>
```

Rippmenüü loomine ise ikka abifunktsioonide failis. Funktsioonile antakse ette kaks parameetrit. Esimene on SQLi SELECT-lause, millest väljastatud tabel annab rippmenüüle valiku id-d ja nähtavad väärtused. Teiseks parameetriks on select-elemendi nimi HTMLi vormis. HTMLi tekst lihtsalt pannakse jupi kaupa kokku ning salvestatakse muutujas $tulemus. Lõpuks antakse sealt ka funktsioonist välja.

```
function looRippMenyy($sqllause, $valikunimi){
   global $yhendus;
   $kask=$yhendus->prepare($sqllause);
   $kask->bind_result($id, $sisu);
   $kask->execute();
   $tulemus="<select name='$valikunimi'>";
   while($kask->fetch()){
     $tulemus.="<option value='$id'>$sisu</option>";
   }
   $tulemus.="</select>";
   return $tulemus;
}
```

Kuna püüame halduslehe enese võimalikult programmikoodist puhta hoida, tuleb abifunktsioone veel mõned juurde kirjutada. Grupi lisamiseks käsklus ühe grupi lisamiseks kaubagruppide tabelisse.

```
function lisaGrupp($grupinimi){
   global $yhendus;
   $kask=$yhendus->prepare("INSERT INTO kaubagrupid (grupinimi)
                    VALUES (?)");
   $kask->bind_param("s", $grupinimi);
   $kask->execute();
}
```

Kauba lisamiseks käsklus kaupade tabelisse rea panekuks. Kusjuures eeldatakse, et parameetrina juba tuleb kaubagrupi_id, kuhu sisse lisatav kaup kuulub. Kuna me eelnev select-valik sai nõnda tehtud, et näha on grupi nimi, kuid serverisse saadetakse kaubagrupi id, siis on nende andmete salvestamine lihtne.

```
function lisaKaup($nimetus, $kaubagrupi_id, $hind){
   global $yhendus;
   $kask=$yhendus->prepare("INSERT INTO
     kaubad (nimetus, kaubagrupi_id, hind)
     VALUES (?, ?, ?)");
   $kask->bind_param("sid", $nimetus, $kaubagrupi_id, $hind);
   $kask->execute();
}
```

Et pärast andmete sisestamist teaks kaubahalduse leht abifunktsioonide alt sobiva käskluse välja kutsuda, tuleb kaubahalduse lehe algusesse lisada kontrollid saabuvate andmete kohta. Kui oli vajutatud nupule nimega grupilisamine, siis jõuab serverisse parameeter sama nimega ning selle järgi teab välja kutsuda grupi lisamise funktsiooni. Samuti tasub käituda kauba lisamise juures. Loodud abifunktsioonile tuleb ette anda kõik lisamiseks vajalikud andmed, mis $_REQUEST-muutuja kaudu sisestusväljadest kohale jõuavad.

```
if(isSet($_REQUEST["grupilisamine"])){
  lisaGrupp($_REQUEST["uuegrupinimi"]);
  header("Location: kaubahaldus.php");
  exit();
}
if(isSet($_REQUEST["kaubalisamine"])){
  lisaKaup($_REQUEST["nimetus"], $_REQUEST["kaubagrupi_id"], $_REQUEST["hind"]);
  header("Location: kaubahaldus.php");
  exit();
}
```

Juurde ka kauba kustutamine. Kusjuures nagu ikka, on viisakas enne küsida, kas ikka tahetakse vastavat kaupa kustutada - et poleks kogemata vajutuse tõttu andmed kaduma läinud. Kõigepealt tuleb siis iga kauba ette vastav kustutusviide teha koos Javaskripti abil küsimisega.

```
<td><a href="kaubahaldus.php?kustutusid=<?=$kaup->id ?>"
       onclick="return confirm('Kas ikka soovid kustutada?')">x</a>
</td>
<td><?=$kaup->nimetus ?></td>
<td><?=$kaup->grupinimi ?></td>
<td><?=$kaup->hind ?></td>
```

Kaubahalduse lehe uuel laadimisel kontrollitakse, et kas äkki on saadetud kaasa kustutusid, mille järgi kauba kustutamine otsustada. Kui jah, siis kutsutakse abifunktsioonide alt välja vastav käsklus.

```
if(isSet($_REQUEST["kustutusid"])){
   kustutaKaup($_REQUEST["kustutusid"]);
}
```

Käsklus ise loogiline - tabelist võetakse ära see rida, mille id kattub saadetud kustutatava kauba id-ga.

```
function kustutaKaup($kauba_id){
   global $yhendus;
   $kask=$yhendus->prepare("DELETE FROM kaubad WHERE id=?");
   $kask->bind_param("i", $kauba_id);
   $kask->execute();
}
```

Ülesandeid

* Tee näide läbi
* Võimalda veebi kaudu lisada maakondi.
* Võimalda veebi kaudu lisada inimeste andmeid, valides rippmenüüst, kus maakonnas nad asuvad.

Andmete muutmine

Mitme seotud andmetabeli pealt kokku ehitatud veebihaldusliidese keerukaimaks kohaks on valitava tunnuse muutmine rippmenüü kaudu - et muutmisel oleks olemasolev valik ette keritud ning võimalik muutus ka õigesti kirja läheks.

Muus osas on muutmine suhteliselt sarnane konspekti algusosas oleva teadetetabeli andmete muutmisega. Vaid SQL-päringud on HTMLi koodist välja viidud ning andmeid näidatakse abifunktsioonid.php-failist tulnud muutujate kaudu.

Kaupade loetelu kuvamisel on kõigepealt iga kauba ees viide kustutamiseks või muutmiseks. Andmete kuvamisel vaadatakse, kas aadressiribalt on saabunud parameeter nimega muutmisid ning kas muudetava kauba id kattub parasjagu näidatava kauba id-ga.

```
          <?php if(isSet($_REQUEST["muutmisid"]) &&
intval($_REQUEST["muutmisid"])==$kaup->id): ?>
```

Kui jah, siis kuvatakse rida tavalisega võrreldes erinevalt, sätitakse andmed sellisele kujule, et neid veebilehelt muuta saab. Nimetus ja hind tulevad nähtavale tekstivälja paigutatult. Kaubagrupp aga paigutatakse sellisesse rippmenüüsse, kus sobiv valik on juba ette keeratud. Selleks tuleb vastavat abifunktsiooni mõnevõrra täiendada, millest edaspidi allpool.

```
 <table>
   <tr>
     <th>Haldus</th>
     <th>Nimetus</th>
     <th>Kaubagrupp</th>
     <th>Hind</th>
   </tr>
   <?php foreach($kaubad as $kaup): ?>
     <tr>
      <?php if(isSet($_REQUEST["muutmisid"]) &&
                    intval($_REQUEST["muutmisid"])==$kaup->id): ?>
       <td>
         <input type="submit" name="muutmine" value="Muuda" />
         <input type="submit" name="katkestus" value="Katkesta" />
         <input type="hidden" name="muudetudid" value="<?=$kaup->id ?>" />
       </td>
       <td><input type="text" name="nimetus" value="<?=$kaup->nimetus ?>" /></td>
       <td><?php
         echo looRippMenyy("SELECT id, grupinimi FROM kaubagrupid",
                           "kaubagrupi_id", $kaup->kaubagrupi_id);
       ?></td>
       <td><input type="text" name="hind" value="<?=$kaup->hind ?>" /></td>
      <?php else: ?>
       <td><a href="kaubahaldus.php?kustutusid=<?=$kaup->id ?>"
              onclick="return confirm('Kas ikka soovid kustutada?')">x</a>
           <a href="kaubahaldus.php?muutmisid=<?=$kaup->id ?>">m</a>
       </td>
       <td><?=$kaup->nimetus ?></td>
       <td><?=$kaup->grupinimi ?></td>
       <td><?=$kaup->hind ?></td>
      <?php endif ?>
     </tr>
   <?php endforeach; ?>
 </table>
```

Kauba andmete näitamise loetellu tuleb parameetriks juurde `kaubagrupi_id`, et oleks võimalik selle abil muutmise ajal rippmenüüst sobivat valikut näidata.

```
function kysiKaupadeAndmed($sorttulp="nimetus", $otsisona=""){
  global $yhendus;
  $lubatudtulbad=array("nimetus", "grupinimi", "hind");
  if(!in_array($sorttulp, $lubatudtulbad)){
    return "lubamatu tulp";
  }
  $otsisona=addslashes(stripslashes($otsisona));
  $kask=$yhendus->prepare("SELECT kaubad.id, nimetus, grupinimi, kaubagrupi_id, hind
     FROM kaubad, kaubagrupid
     WHERE kaubad.kaubagrupi_id=kaubagrupid.id
      AND (nimetus LIKE '%$otsisona%' OR grupinimi LIKE '%$otsisona%')
     ORDER BY $sorttulp");
  $kask->bind_result($id, $nimetus, $grupinimi, $kaubagrupi_id, $hind);
  $kask->execute();
  $hoidla=array();
  while($kask->fetch()){
    $kaup=new stdClass();
    $kaup->id=$id;
    $kaup->nimetus=htmlspecialchars($nimetus);
    $kaup->grupinimi=htmlspecialchars($grupinimi);
    $kaup->kaubagrupi_id=$kaubagrupi_id;
    $kaup->hind=$hind;
    array_push($hoidla, $kaup);
  }
  return $hoidla;
}
```

Rippmenüüle tuleb juurde täiendav parameeter nimega $valitudid. Nagu eelnevalt otsimise juures nii ka siin võimaldab lisandunud parameetrile vaikeväärtuse jätmine kasutada funktsiooni samaaegselt nii senini kehtinud kahe parameetriga kohas (SELECT-lause ning valiku HTMLi sees olev nimi) kui ka uues kohas, kus antakse ette id, millist rida andmete seast ette kerida. Kui id-

parameeter puudub, siis antakse sellele vaikimisi väärtuseks tühi tekst. Ning kui sellise väärtusega id-d pole (üldjuhul ei tohiks olla), siis lihtsalt eraldi ei valitagi midagi välja, rippmenüü tuleb nähtavale nii nagu ennegi. Kui aga funktsioonile anti ettekeritav id ning sarnane id leidub ka näidatavate andmete seas, siis lisatakse vastavale valikule parameeter `selected='selected'`. Koodi mugavamaks kirjutamiseks on see toimetus jagatud kolme ritta. Vaikimisi pannakse lisandit hoidvale muutujale väärtuseks tühi tekst, st. et selle muutuja väärtuse lisamine ei muuda väljundit kuhugi. Järgmise tingimusega vaadatakse, et kas sinna on selected-atribuut põhjust sisse kirjutada. Ning kui kirjutati, siis jõuab see kolmandal real ilusti ka vastava option-rea kirjeldusse.

```
      $lisand="";
      if($id==$valitudid){$lisand=" selected='selected'";}
      $tulemus.="<option value='$id' $lisand >$sisu</option>";
```

Alamprogramm uuel kujul tervikuna:

```
function looRippMenyy($sqllause, $valikunimi, $valitudid=""){
   global $yhendus;
   $kask=$yhendus->prepare($sqllause);
   $kask->bind_result($id, $sisu);
   $kask->execute();
   $tulemus="<select name='$valikunimi'>";
   while($kask->fetch()){
     $lisand="";
     if($id==$valitudid){$lisand=" selected='selected'";}
     $tulemus.="<option value='$id' $lisand >$sisu</option>";
   }
   $tulemus.="</select>";
   return $tulemus;
}
```

Ning nende toimetuste tulemusena tekib siis valitud rida juba muudetaval kujul ekraanile. Muutused küll veel kuhugile ei jõua, aga eks see tuleb siis järgmise sammuna üles tähendada.

Haldus		Nimetus	Kaubagrupp	Hind
Muuda	Katkesta	ahjutellis	tellised	8.20
x m		bituumenrull	katusematerjal	525.00
x m		fassaaditellis	tellised	7.50

Kauba andmete muutmiseks lisatakse taas sobiv funktsioon. Kindlasti tuleb muutmise juurest kaasa saata muudetava kauba `id`. Selleks oli ennist vormi sees rida

```
<input type="hidden" name="muudetudid" value="<?=$kaup->id ?>" />
```

kus see `id` kaasa pandi.

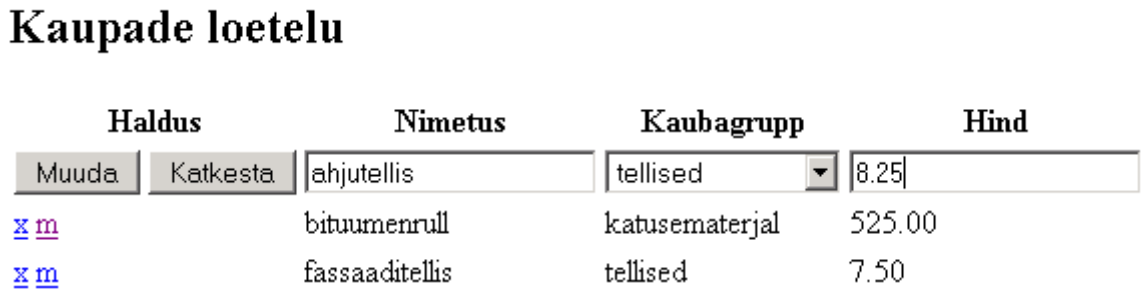
Edasi koostatakse abifunktsioonide alla UPDATE-lause, kus id-numbriga määratud real vanad andmed uutega üle kirjutatakse.

```
function muudaKaup($kauba_id, $nimetus, $kaubagrupi_id, $hind){
   global $yhendus;
   $kask=$yhendus->prepare("UPDATE kaubad SET nimetus=?, kaubagrupi_id=?, hind=?
                    WHERE id=?");
   $kask->bind_param("sidi", $nimetus, $kaubagrupi_id, $hind, $kauba_id);
   $kask->execute();
}
```

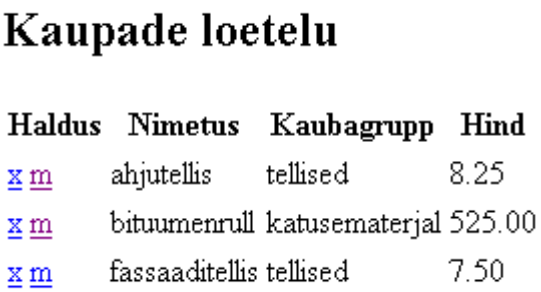
Kaupade lehe juures tuleb algusesse juurde kontroll - et kui vajutati muutmisnupule, siis käivitatakse muutmisfunktsioon koos ette antud uute andmetega.

```
  if(isSet($_REQUEST["muutmine"])){
     muudaKaup($_REQUEST["muudetudid"], $_REQUEST["nimetus"],
                          $_REQUEST["kaubagrupi_id"], $_REQUEST["hind"]);
  }
```

Ning võibki uue hinna kirjutada ja muudki parandused vajadusel siise viia.



Muutmisnupu vajutamise järgsel lehe avamisel ongi uuel kujul andmed siis nähtavad.



Järeltegemise hõlbustamiseks siis halduseks vajalike failide koodid tervikuna.

kaubahaldus.php

```
<?php
  require("abifunktsioonid.php");
  if(isSet($_REQUEST["grupilisamine"])){
    lisaGrupp($_REQUEST["uuegrupinimi"]);
    header("Location: kaubahaldus.php");
    exit();
  }
  if(isSet($_REQUEST["kaubalisamine"])){
    lisaKaup($_REQUEST["nimetus"], $_REQUEST["kaubagrupi_id"], $_REQUEST["hind"]);
    header("Location: kaubahaldus.php");
    exit();
  }
  if(isSet($_REQUEST["kustutusid"])){
     kustutaKaup($_REQUEST["kustutusid"]);
  }
  if(isSet($_REQUEST["muutmine"])){
     muudaKaup($_REQUEST["muudetudid"], $_REQUEST["nimetus"],
                              $_REQUEST["kaubagrupi_id"], $_REQUEST["hind"]);
  }
  $kaubad=kysiKaupadeAndmed();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
  <head>
      <title>Kaupade leht</title>
      <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  </head>
  <body>
   <form action="kaubahaldus.php">
     <h2>Kauba lisamine</h2>
     <dl>
       <dt>Nimetus:</dt>
       <dd><input type="text" name="nimetus" /></dd>
       <dt>Kaubagrupp:</dt>
       <dd><?php
         echo looRippMenyy("SELECT id, grupinimi FROM kaubagrupid",
                           "kaubagrupi_id");
       ?>
       </dd>
       <dt>Hind:</dt>
       <dd><input type="text" name="hind" /></dd>
     </dl>
     <input type="submit" name="kaubalisamine" value="Lisa kaup" />
     <h2>Grupi lisamine</h2>
     <input type="text" name="uuegrupinimi" />
     <input type="submit" name="grupilisamine" value="Lisa grupp" />
   </form>
   <form action="kaubahaldus.php">
     <h2>Kaupade loetelu</h2>
     <table>
       <tr>
         <th>Haldus</th>
         <th>Nimetus</th>
         <th>Kaubagrupp</th>
         <th>Hind</th>
       </tr>
       <?php foreach($kaubad as $kaup): ?>
         <tr>
          <?php if(isSet($_REQUEST["muutmisid"]) &&
                           intval($_REQUEST["muutmisid"])==$kaup->id): ?>
           <td>
             <input type="submit" name="muutmine" value="Muuda" />
             <input type="submit" name="katkestus" value="Katkesta" />
             <input type="hidden" name="muudetudid" value="<?=$kaup->id ?>" />
           </td>
           <td><input type="text" name="nimetus" value="<?=$kaup->nimetus ?>" /></td>
           <td><?php
             echo looRippMenyy("SELECT id, grupinimi FROM kaubagrupid",
                                        "kaubagrupi_id", $kaup->kaubagrupi_id);
           ?></td>
           <td><input type="text" name="hind" value="<?=$kaup->hind ?>" /></td>
          <?php else: ?>
           <td><a href="kaubahaldus.php?kustutusid=<?=$kaup->id ?>"
                  onclick="return confirm('Kas ikka soovid kustutada?')">x</a>
              <a href="kaubahaldus.php?muutmisid=<?=$kaup->id ?>">m</a>
           </td>
           <td><?=$kaup->nimetus ?></td>
           <td><?=$kaup->grupinimi ?></td>
           <td><?=$kaup->hind ?></td>
          <?php endif ?>
         </tr>
       <?php endforeach; ?>
     </table>
   </form>
  </body>
</html>
```

ning abifunktsioonid.php

```
<?php
  $yhendus=new mysqli("localhost", "jaagup", "xxxxxx", "jaagup");
```

```
  function kysiKaupadeAndmed($sorttulp="nimetus", $otsisona=""){
    global $yhendus;
    $lubatudtulbad=array("nimetus", "grupinimi", "hind");
    if(!in_array($sorttulp, $lubatudtulbad)){
      return "lubamatu tulp";
    }
    $otsisona=addslashes(stripslashes($otsisona));
    $kask=$yhendus->prepare("SELECT kaubad.id, nimetus, grupinimi, kaubagrupi_id, hind
       FROM kaubad, kaubagrupid
       WHERE kaubad.kaubagrupi_id=kaubagrupid.id
        AND (nimetus LIKE '%$otsisona%' OR grupinimi LIKE '%$otsisona%')
       ORDER BY $sorttulp");
    //echo $yhendus->error;
    $kask->bind_result($id, $nimetus, $grupinimi, $kaubagrupi_id, $hind);
    $kask->execute();
    $hoidla=array();
    while($kask->fetch()){
      $kaup=new stdClass();
      $kaup->id=$id;
      $kaup->nimetus=htmlspecialchars($nimetus);
      $kaup->grupinimi=htmlspecialchars($grupinimi);
      $kaup->kaubagrupi_id=$kaubagrupi_id;
      $kaup->hind=$hind;
      array_push($hoidla, $kaup);
    }
    return $hoidla;
  }

  /**
  * Luuakse HTML select-valik, kus v6etakse v22rtuseks sqllausest tulnud
  * esimene tulp ning n2idatakse teise tulba oma.
  */

  function looRippMenyy($sqllause, $valikunimi, $valitudid=""){
     global $yhendus;
     $kask=$yhendus->prepare($sqllause);
     $kask->bind_result($id, $sisu);
     $kask->execute();
     $tulemus="<select name='$valikunimi'>";
     while($kask->fetch()){
       $lisand="";
       if($id==$valitudid){$lisand=" selected='selected'";}
       $tulemus.="<option value='$id' $lisand >$sisu</option>";
     }
     $tulemus.="</select>";
     return $tulemus;
  }

/*
  function looRippMenyy($sqllause, $valikunimi){
     global $yhendus;
     $kask=$yhendus->prepare($sqllause);
     $kask->bind_result($id, $sisu);
     $kask->execute();
     $tulemus="<select name='$valikunimi'>";
     while($kask->fetch()){
       $tulemus.="<option value='$id'>$sisu</option>";
     }
     $tulemus.="</select>";
     return $tulemus;
  }
*/

  function lisaGrupp($grupinimi){
     global $yhendus;
     $kask=$yhendus->prepare("INSERT INTO kaubagrupid (grupinimi)
                      VALUES (?)");
     $kask->bind_param("s", $grupinimi);
     $kask->execute();
  }
```

```
  function lisaKaup($nimetus, $kaubagrupi_id, $hind){
     global $yhendus;
     $kask=$yhendus->prepare("INSERT INTO
       kaubad (nimetus, kaubagrupi_id, hind)
       VALUES (?, ?, ?)");
     $kask->bind_param("sid", $nimetus, $kaubagrupi_id, $hind);
     $kask->execute();
  }

  function kustutaKaup($kauba_id){
     global $yhendus;
     $kask=$yhendus->prepare("DELETE FROM kaubad WHERE id=?");
     $kask->bind_param("i", $kauba_id);
     $kask->execute();
  }

  function muudaKaup($kauba_id, $nimetus, $kaubagrupi_id, $hind){
     global $yhendus;
     $kask=$yhendus->prepare("UPDATE kaubad SET nimetus=?, kaubagrupi_id=?, hind=?
                      WHERE id=?");
     $kask->bind_param("sidi", $nimetus, $kaubagrupi_id, $hind, $kauba_id);
     $kask->execute();
  }

  //---------------
  if( array_pop(explode("/", $_SERVER["PHP_SELF"]))=="abifunktsioonid.php"):
?>
<pre>
<?php
  print_r(kysiKaupadeAndmed("hind", "fass\\aad"));
?>
</pre>
<?php endif ?>
```

Ülesandeid

* Tee näited läbi
* Võimalda muuta inimeste andmeid, kaasa arvatud maakonda, kus inimene paikneb.

* Koosta kommenteeritavate uudistega veebilehestik. Administraatorilehel saab lisada uudiseid.
* Vaatajalehel näeb uudiseid ning neid saab kommenteerida. Kommentaaride tabelis on viide uudise id-le.

* Administraatorilehel on võimalus sobimatute kommentaaride märkimiseks. Kommentaaride lehel näidatakse sellel kohal teadet, et ”kommentaar oli sobimatu”.