Duomenų bazė

DUOMENŲ PERŽIŪRA

SELECT sakinys

SQL sakinys SELECT yra pagrindinis sakinys DB-je esantiems duomenims peržiūrėti. Šis sakinys turi daug įvairiausių variantų ir galimybių. Išsamiai šio sakinio sintaksei pateikti prireiktų keleto puslapių. Sakinį bendriausiu atveju galima užrašyti taip:

SELECT [DISTINCT] <stulpelių vardai>
FROM <lentelių vardai>
[WHERE <paieškos sąlyga>]
[GROUP BY <stulpelių vardai> [HAVING <Paieškos sąlyga>]]
[ORDER BY <stulpelių vardai>] .

Įvykdžius šį sakinį, DBVS suformuoja ir pateikia vartotojui užklausos rezultatą – laikiną lentelę, kuri egzistuoja tik užklausos rezultato peržiūros metu. Sistemos vartotojas rezultatą gali pamatyti monitoriaus ekrane ar apdoroti jį programoje.

Nesunku pastebėti, kad pati paprasčiausia užklausa atrodo taip:

SELECT <stulpelių vardai> FROM <lentelės vardas>.

SELECT sakinys

SELECT užklausą sudaro kelios dalys.

Pirmoji dalis, iš karto po žodžio SELECT, yra laukų sąrašas.

Pavyzdžiui:

```
SELECT commission, employee_number FROM sales_rep WHERE surname='Gordimer';
```

SELECT sakinys

Taip pat galima panaudoti ir pakaitos simbolį (*), kad būtų pateikti visi laukai, kaip čia:

```
SELECT * FROM sales_rep WHERE surname='Gordimer';
```

Pakaitos simbolis * atitinka visus lentelės laukus. Taigi prieš tai pateiktame pavyzdyje pateikiami visi keturi laukai tokia pačia tvarka, kokia jie išrikiuoti lentelės struktūroje .

SELECT sakinys

Vienodų eilučių galima išvengti panaudojant bazinį žodį DISTINCT. Sakinį perrašę taip:

- SELECT DISTINCT Išsilavinimas FROM Vykdytojai

Įtraukus DISTINCT sakinį vienodos eilutės pašalinamos. Neįtraukus į užklausą bazinio žodžio DISTINCT, užklausos rezultate galimos vienodos eilutės, tiksliau, vienodos eilutės nėra pašalinamos.

Apskaičiuojami stulpeliai

Kai kurie stulpeliai gali būti apskaičiuojami. Tarkime, kad lentelėje Projektai projektų trukmė nurodyta mėnesiais, o mes tą laiką norime sužinoti dienomis.

Paprastumo dėlei tarkime, kad kiekviename mėnesyje yra 30 dienų. Uždavinį išspręsime sakiniu:

```
SELECT Pavadinimas, Trukmė * 30 FROM Projektai .
```

Šiame sakinyje yra pavartotas skaičius 30 – tai skaitinių duomenų konstanta. Skaičiai SQL sakiniuose rašomi be jokių papildomų atskyrėjų. Šios užklausos rezultatas yra laikina lentelė, turinti du stulpelius.

Stulpelių pavadinimai

Kai užklausos SELECT frazėje vartojamas reiškinys, rezultato stulpelio pavadinimas gali būti dviprasmiškas.

Tokiais atvejais sistema, vaizduodama užklausos rezultatą, stulpeliui suteikia tarnybinį pavadinimą, kuris atitinka stulpelio eilės numerį užklausoje (antrajam stulpeliui suteikiamas vardas “2”). Tai nevisada priimtina. Vartotojas gali pats nurodyti norimą stulpelio pavadinimą, pavartojant bazinį žodį **AS**.

Suteikiamas stulpeliui pavadinimas turi tenkinti tuos pačius reikalavimus, kaip ir duomenų bazės stulpelio pavadinimas.

Norint stulpeliui suteikti pavadinimą, kuriame būtų vienas ar keli specialieji simboliai, būtina stulpelio pavadinimą rašyti tarp kabučių:

SELECT Pavadinimas, Trukmė * 30 AS “Trukmė dienomis” FROM Projektai .

Sąlygos sakinys

SELECT sakinio dalis, esanti po WHERE, yra vadinama sąlygos *sakiniu.* Šis sakinys labai lankstus ir jame gali būti daugybė įvairių tipų sąlygų. Pažvelkite į šį pavyzdį:

SELECT * FROM sales_rep WHERE commission>10 OR surname='Rive' AND first_name='Sol';

Vardinės konstantos

Užklausose jau vartojome pačius paprasčiausius reiškinius: konstantas, stulpelių pavadinimus ir aritmetinius reiškinius.

Vardinės konstantos (sisteminiai pseudokintamieji) - tai reikšmės, kurias saugo DBVS ir kurias galima vartoti SQL sakiniuose. Paprastai, tai išorinė DBVS požiūriu informacija, žyminti einamąją sisteminę datą (CURRENT DATE**)**, laiką **(**CURRENT TIME**)**, datą ir tikslų laiką (CURRENT TIMESTAMP), sistemos vartotojo vardą (USER) ir pan. Kai kuriose DBVS, sisteminės reikšmės pasiekiamos ne per vardines konstantas, bet kviečiant specialias funkcijas

Funkcijos

Funkcija - tai operacija, nusakoma funkcijos vardu ir apskliaustais lenktiniais skliaustais argumentais, kurie tarpusavyje atskiriami kableliais (atskiru atveju argumentų gali ir nebūti).

Funkcija visuomet grąžina tam tikrą rezultatą (tai gali būti ir specialioji reikšmė NULL).

Funkcijos yra skirtomos į agregatines (stulpelių) ir skaliarines.

Agregatinės f-jos

Agregatinės funkcijos - tai funkcijos, kurias galima pritaikyti eilučių aibėms: visoms eilutėms, tenkinančioms frazėje WHERE nurodytą sąlygą, arba eilučių grupei, apibrėžiamai fraze GROUP.

Agregatinės funkcijos bet kokiam eilučių rinkiniui apskaičiuoja vieną reikšmę. Šios rūšies funkcijos dažnai vartojamos SELECT frazėje, nurodant stulpelį, pagal kurio reikšmes apskaičiuojamas funkcijos rezultatas.

Agregatinės funkcijos vartojamos ir frazėje HAVING, kai funkcijos reikšmė skaičiuojama kiekvienai eilučių grupei.

Agregatinės f-jos

Keletas dažnai vartojamų agregatinių funkcijų:

Agregatinė funkcija	Rezultatas
SUM([DISTINCT] <reiškinys>)	(Skirtingų) ne NULL reikšmių suma
AVG([DISTINCT] <reiškinys>)	(Skirtingų) ne NULL reikšmių vidurkis
COUNT([DISTINCT] <reiškinys>)	(Skirtingų) ne NULL reikšmių kiekis
COUNT(*)	Eilučių kiekis aibėje
MAX(<reiškinys>)	Maksimali reikšmė
MIN(<reiškinys>)	Minimali reikšmė

Jei visos funkcijoje nurodyto stulpelio reikšmės yra NULL, arba stulpelis yra tuščias, tai funkcijų SUM, AVG, MIN, MAX rezultatas yra NULL, o funkcijos COUNT – nulis.

Agregatinės f-jos

Visų vykdytojų skaičius arba, tiksliau, eilučių kiekis lenteleje Vykdytojai gali būti sužinomas sakiniu:

SELECT COUNT(*) FROM Vykdytojai.

Visų vykdytojų, dalyvaujančių bent viename projekte, skaičius - sakiniu:

- SELECT COUNT**(**DISTINCT Vykdytojas**)** FROM Vykdymas.

Agregatinės f-jos

Bendrą kiekį valandų, kurias vykdytojas Nr. 1 skiria visiems projektams, galima sužinoti įvykdžius užklausą:

```
SELECT SUM(Valandos) AS “Vykdytojo Nr. 1 valandos”
    FROM Vykdymas
    WHERE Vykdytojas = 1
```

Skaliarinės funkcijos

Skaliarinės funkcijos argumentas visuomet yra viena reikšmė.

Funkcija gali turėti ir kelis argumentus, tačiau kiekvienas argumentas – viena reikšmė, o ne reikšmių aibė, kaip agregatinėje funkcijoje.

Vartojant skaliarines funkcijas frazėje WHERE, funkcijos rezultatas apskaičiuojamas tikrinant paieškos sąlygą kiekvienai lentelės eilutei atskirai.

Šios rūšies funkcijos dažnai vartojamos ir SELECT frazėje, kuomet rezultatas skaičiuojamas ne visoms eilutėms iš karto, kaip yra agregatinių funkcijų atveju, o kiekvienai eilutei atskirai.

Skaliarinės funkcijos

DBVS leidžia vartoti gana daug skaliarinių funkcijų. Paminėsime tik keletą iš jų:

- DAY(<data>) – diena (skaičius nuo 1 iki 31) datoje,
- MONTH(<data>) – mėnesis datoje,
- YEAR(<data>) – metai datoje,
- LENGTH(<simbolių eilutė>) – simbolių eilutės ilgis,
- SUBSTR(<simbolių eilutė>, <pradžia>, <ilgis>) – simbolių eilutės fragmentas

ir pan.

Skaliarinės funkcijos

Iš konstantų, kreipinių į funkcijas bei vardų, žyminčių DB objektus, panaudojant operacijas, galima sudaryti paprasčiausius reiškinius.

SQL leidžia naudoti šias operacijas:

+ (sudėtis),

- (atimtis),

* (daugyba),

/(dalyba),

|| (apjungimas).

Jei bent vienas iš pateiktų operacijų argumentų yra NULL, tai ir rezultatas yra NULL.

Skaliarinės funkcijos

Su tekstiniais duomenimis galima atlikti tik dviejų simbolių eilučių nuoseklų apjungimą į vieną (||).

Su skaičiais, kaip įprasta, galima atlikti visas aritmetines operacijas.

Sudėtį ir atimtį galima atlikti ir su datos bei laiko duomenimis, tiksliau, turimą reikšmę galima padidinti, sumažinti bei rasti dviejų reikšmių skirtumą.

Prie datos galima pridėti ar iš jos atimti tam tikrą sveiką skaičių dienų, mėnesių ar metų. Laiką galima pakeisti laiko vienetais.

Predikatai

Reiškiniai gali būti predikatų argumentais (operandais).

Predikatas - tai sąlyga lentelės eilutei ar eilučių grupei, kuri gali būti teisinga, neteisinga arba neapibrėžta.

SQL leidžiamos tokios palyginimo operacijos: =, <, <=, >, >=, <>.

Jei palyginimo operacijose vienas iš operandų yra NULL**,** tai predikato rezultatas yra neapibrėžtas. Palyginimo operacijose operandais gali būti ne tik skaitiniai duomenys, bet ir tekstiniai bei datos ir laiko duomenys.

Predikatai

Prie paprasčiausių predikatų priskiriama:

- x BETWEEN y AND z - rezultatas teisingas tik tuomet, kai x reikšmė yra tarp y ir z, t.y. kai x >= y ir x <= z;
- x NOT BETWEEN y AND z - rezultatas teisingas tik tuomet, kai x nėra tarp y ir z, t.y. kai x < y arba x > z;
- x IN (y1, y2,..., yn) - rezultatas teisingas tik tuomet, kai x reikšmė sutampa bent su viena iš reikšmių y1, y2,..., yn;
- x NOT IN (y1, y2,..., yn) - rezultatas teisingas tik tuomet, kai x nesutampa nei su viena iš reikšmių y1, y2, ..., yn;
- x IS NULL - rezultatas yra teisingas tik tuomet, kai reiškinio x reikšmė yra NULL;
- x IS NOT NULL - rezultatas yra teisingas tik tuomet, kai reiškinio x reikšmė nėra NULL.

Predikatai

x LIKE y - rezultatas teisingas tik tuomet, kai simbolių eilutė x yra “panaši” į simbolių eilutę y.

x NOT LIKE y - rezultatas yra teisingas tik tuomet, kai simbolių eilutė x nėra panaši į simbolių eilutę y. Eilutėje y gali būti pavartoti tie patys simboliai kaip ir predikate LIKE;

Jei tarkime pamenate, kad pavardė prasideda *Petr,* galite įvykdyti tokią užklausą:

SELECT * FROM darbuotojai WHERE pavarde LIKE ' *Petr* % ';

Šablono atitikimas: LIKE ir %

% - tai pakaitos simbolis, panašiai kaip ir simbolis *, tačiau skirtas naudoti tik SELECT sąlygose. Jis reiškia 0 arba daugiau simbolių.

Pakaitos simboli galite naudoti kiek norite kartų, o tai leidžia ivykdyti užklausas, tokias kaip ši:

SELECT * FROM pardavimai WHERE pavarde LIKE '%e%'

Pavyzdžiai

Pavyzdžiui, projektų, kurių pavadinime yra frazė “apskaita”, jų svarba yra vidutinė ar didelė, ir kurie turėjo būti pabaigti iki šiandien, pavadinimus, vykdymo pradžios bei pabaigos datas galima sužinoti tokia užklausa:

SELECT *Pavadinimas, Pradžia, Pradžia+Trukmė* MONTHS AS *Pabaiga*

FROM *Projektai*

WHERE *Pavadinimas* LIKE *‘%*apskaita%’ AND

Svarba IN ('Vidutinė', 'Didelė')

Informacija apie vykdytojus – informatikus arba vykdytojus, baigusius Vilniaus universitetą (nepriklausomai nuo kvalifikacijos) ir turinčius aukštesnę nei trečią kategoriją, bus pateikta įvykdžius užklausą:

SELECT * FROM *Vykdytojai* WHERE *Kvalifikacija* = 'Informatikas' OR (*Išsilavinimas* = 'VU' AND *Kategorija* > 3).

Rikiavimas

Kitas naudingas ir dažnai naudojamas sakinys leidžia rikiuoti rezultatus, Būtų naudinga pamatyti abėcėles tvarka išrikiuotą darbuotojų sąrašą, ir norėdami ji išvysti, galite panaudoti ORDER BY sakinį:

```
SELECT * FROM pardavimai ORDER BY pavarde;
```

Šis sąrašas nebus visiškai teisingas, jeigu norite išrikiuoti pagal vardus, kadangi *Pavardės gali sutapti, o vardai skirtis.* Norėdami tai pataisyti, turite išrikiuoti ir pagal vardą tuo atveju, jei pavardės sutampa.

Kad tai padarytumėte, turite įvykdyti tokią užklausą:

```
SELECT * FROM pardavimai ORDER BY pavarde, vardas;
```

Rikiavimas

Norėdami išrikiuoti atvirkščia tvarka (mažėjančia), turite panaudoti reikšmini žodi DESC. Kita užklausa grąžins visus įrašus, išrikiuotus pagal uždirbtus komisinius, pradedant didžiausiu ir baigiant mažiausiu:

```
SELECT * FROM pardavimai ORDER BY komisiniai DESC;
```

Rikiavimas

Gali prireikti, kad trys darbuotojai, gaunantys vienodus komisinius, būtų toliau Rikiuojami pagal pavardę ir vardą.

Norėdami tai padaryti, galite panaudoti reikšmini žodi ASC. Ir nors nėra griežtai reikalaujamas, kadangi tai yra numatytoji rikiavimo tvarka, šio reikšminio žodžio dėka užklausa tampa suprantamesnė:

```
SELECT * FROM pardavimai ORDER BY komisiniai DESC, pavarde ASC, vardas ASC;
```

Rezultatų skaičiaus ribojimas

Realioje duomenų bazėje gali būti ne vienas tūkstantis įrašų ir jūs vienu metu nenorėsite pamatyti jų visų.

Todėl MySQL leidžia panaudoti LIMIT sąlygą.

LIMIT nėra standartiniame SQL, o tai reiškia, kad negalėsite jo taip pat panaudoti ir visose kitose duomenų bazėse, tačiau tai naudingas ir galingas MySQL įrankis.

SELECT vardas, pavarde, komisiniai FROM pardavimai ORDER BY komisiniai DESC LIMIT 1:

Rezultatų skaičiaus ribojimas

LIMIT sakinys leidžia ne tik išvesti ribotą skaičių įrašų, skaičiuojant nuo pradžios ar pabaigos.

Jūs galite nurodyti MySQL, nuo kelintos eilutės skaityti įrašus ir kiek eilučių rodyti. Jeigu LIMIT sąlygojė yra du skaičiai, pirmasis žymi praleidžiamų, o antrasis - rodomų eilučių skaičių.

Tolesniame pavyzdyje išvedamas antrasis įrašas, kuomet duomenys išrikiuoti mažėjančia tvarka:

```
SELECT vardas, pavarde, komisiniai FROM pardavimai ORDER BY komisiniai DESC LIMIT 1,1
```

Rezultatų skaičiaus ribojimas

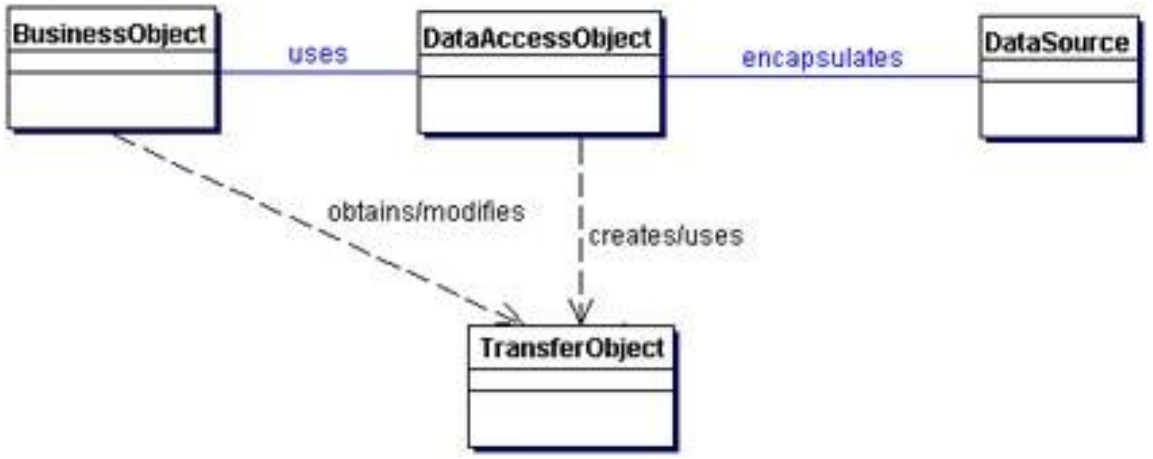
LIMIT yra dažnai naudojamas paieškos varikliuose, kurie naudoja, pavyzdžiui, viename puslapyje norint pateikti tik 10 rezultatų.

Tokiu atveju pirmojo puslapio rezultatai naudotų LIMIT 0,10, antrojo LIMIT 10,10 ir t.t.

DAO projektavimo šablonas

DAO (angl. Data Access Object) – tai vienas iš standartinių J2EE architektūrinių modelių (angl. pattern).

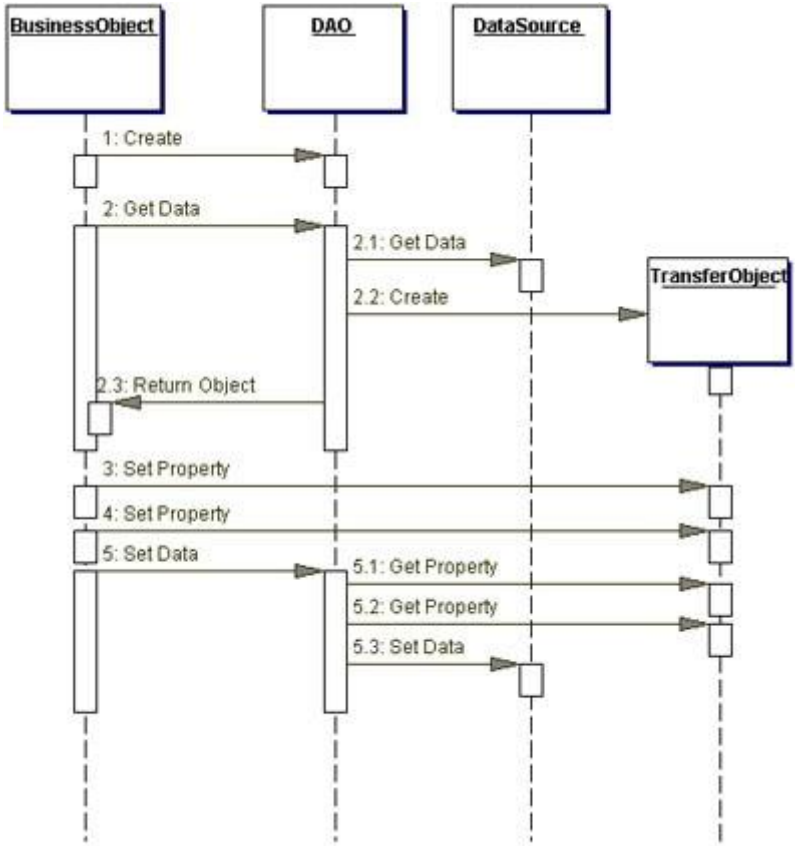
DAO dažniausiai naudojamas tam, kad atskirti žemo lygio (angl. low-level) duomenų priėjimo logiką nuo aukšto lygio verslo logikos (angl. high-level business logic).



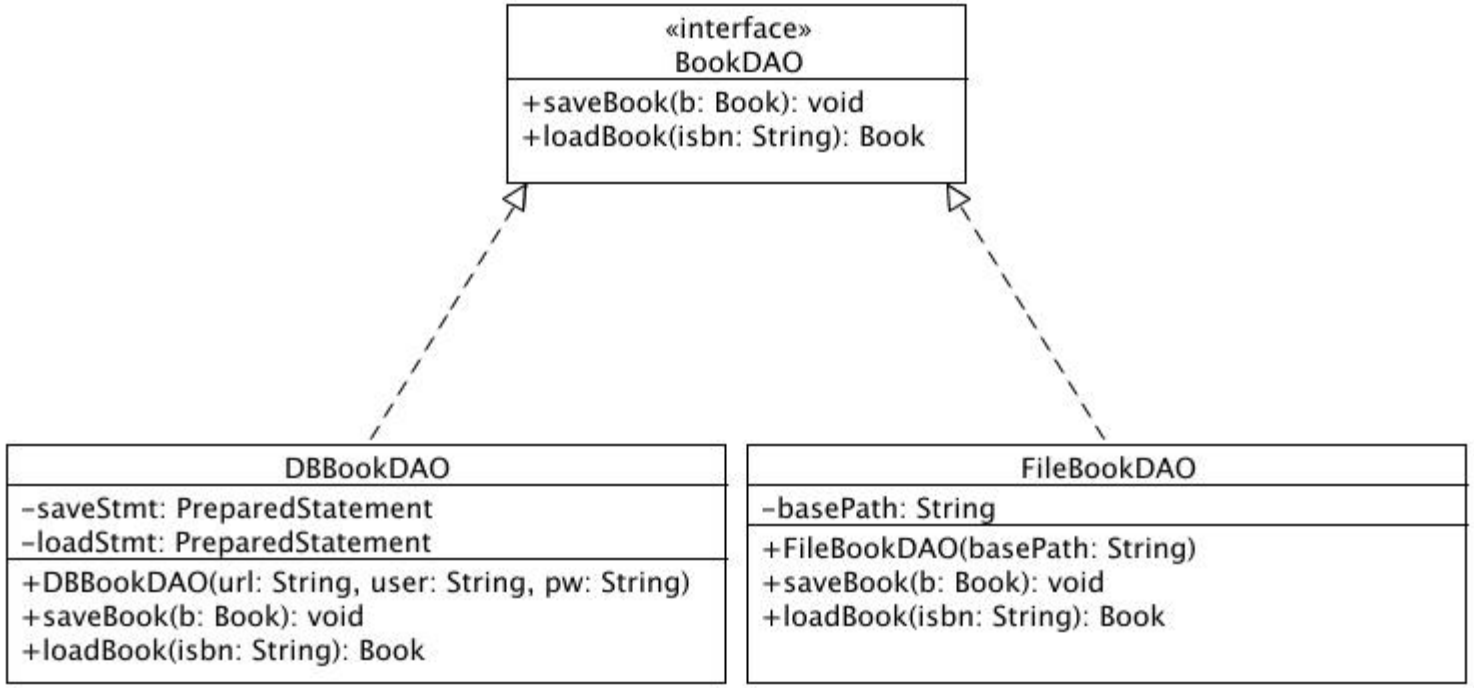
DAO taikymas

Panaudojant DAO dirbama su paprastesne sąsaja.

Taip pat DAO paslepia duomenų šaltinio realizaciją nuo klientų. Tai leidžia pridėti lankstumą serveriui ir kitiems programuotojams, yra nebūtina žinoti .



DAO taikymo pavyzdys



DB įrašų atvaizdavimas

Norėdami atvaizduoti įrašus iš DB panaudodami DAO architektūrą turėtumėme atlikti tokią seką:

1. Sukurti Objekto klasę
2. Sukurti Objekto klasės užkrovimo klasės interfeisą
3. Sukurti Objekto užkrovimo klasę (DAO)
4. Sukurti servletą kuris sukuria Objekto užkrovimo klasės objektą ir paima duomenis
5. Sukurti JSP failą kuriame bus padarytas atvaizdavimas, jį rodys servletas

1. Sukurti Objekto klasę

```
public class Countrie {
  private String code;
  private String name;
  private int area;
  public Countrie(String code, String name,  int area) {
    this.code = name;
    this.name = name;
    this.area = area;
  }
}
```

2. Sukurti Objekto klasės užkrovimo klasės interfeisą

```
public interface CountrieDAO {
   public List<Countrie> getCountries();
   public Countrie getCountrie(String countrie);
}
```

3. Sukurti Objekto užkrovimo klasę (DAO)

```
public class CountrieDB implements CountrieDAO {
private DataSource dataSource;

CountrieDB(DataSource dataSource) {

  this.dataSource= dataSource;

}
@Override
public List<Countrie> getCountries()  {
  List<Countrie> countries = new LinkedList<>();
  Connection connection;
  Statement statement;
  ResultSet rs;
try {
  connection = dataSource.getConnection();
  statement = connection.createStatement();
  String sql = "SELECT * FROM countries;";
  rs = statement.executeQuery(sql);
  while (rs.next()) {
      String code = rs.getString("name");
      String name = rs.getString("name");
      int area = rs.getInt("area");
      countries.add(
        new Countrie(code, name, area));
    }
    rs.close();
    statement.close();
    connection.close();
  } catch (Exception e) {
    e.printStackTrace();
  }

  return countries;
}

}
```

4. Sukurti servletą kuris sukuria Objekto užkrovimo klasės objektą ir paima duomenis

```
@WebServlet("/Countries")
public class Countries extends HttpServlet {

  @Resource (name="jdbc/naujas")
  private DataSource dataSource;
  private CountrieDB countrieDB;

  @Override
  public void init() throws ServletException {
    super.init();
    this.countrieDB=new CountrieDB(dataSource);
  }
  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    // Šalių atvaizdavimas
  }
}
```

5. Sukurti JSP failą kuriame bus padarytas atvaizdavimas, jį rodys servletas

Perduodame duomenis iš servleto ir juos atvaizduojame JSP faile

Parametrizuotų užklausų vykdymas

```
connection = dataSource.getConnection();
String sql = "SELECT * FROM countries WHERE code=?";
statement=connection.prepareStatement(sql);
statement.setString(1, "LTU");
rs = statement.execute();
```