

Sissejuhatus andmebaasidesse – tunniülesanne

Tunnitöö eesmärk on tutvuda andmebaaside, SQL päringukeele ja andmebaasihalduri liidesega. Tööülesannete jaoks on olemas järgnevate ühendusparameetritega PostgreSQL andmebaas:

```
Host: ewis.pld.ttu.ee
Andmebaas: ablid
Port: 5432
Kasutaja: student
Parool: iag0582
```

Tunniülesande lahendamiseks on vajalik tabel nimega **'joe_andmed'**, mis sisaldab endas Eesti jõgesid ja tabel nimega **'seire_jaamad'**.

Tabeli **'joe_andmed'** väljadest on olulised jõe pikkus (*pikkus_km*) ning jõe algus- ja lõpp-punkti geograafilised koordinaadid (*lon1*, *lat1* – algus; *lon2*, *lat2* – lõpp). Igal jõel ja jões toimuvate muutuste jälgimiseks mõeldud seirejaamal on oma identifikaator. Tabelid **'joe_andmed'** ja **'seire_jaamad'** on omavahel seotud identifikaatori **'id_jogi'** abil – st iga seirejaam on seotud ühe kindla jõega, mille ääres ta paikneb.

joe_andmed		seire_jaamad	
id_jogi	int4[10]	id_jaam	
id_suubla	int4[10]	jaama_nimi	
id_peajogi	int4[10]	nimeke	
id_vesikond	int4[10]	suue_km	
id_avk	varchar[6]	valgla_km2	
avk_nimi	varchar[13]	l_est_lon_x	
joenimi	varchar[15]	l_est_lat_y	
tyyp	int4[10]	lavendi_nimi	
suubjark	int4[10]	suue_m	
suudmest_km	numeric[4,1]	kommentaar	
pikkus_km	numeric[4,1]	< 1	141 rows 1 >
lon1	numeric[8,6]		
lat1	numeric[8,6]		
lon2	numeric[8,6]		
lat2	numeric[8,6]		
< 0	1,755 rows		1 >

Ülesanded

1. Kirjuta programm, mis looks ühenduse eeltoodud andmebaasiga ja väljastaks 5 pikimat jõge.
2. Pärida jõed, mille *tyyp* väli vastab matrikli viimasele numbrile ning mis on pikemad kui 30 km.
3. Leida eelmises punktis leitud jõgedel asetsevad seirejaamad.
 - Võtmesõnadeks **'inner join'** ja **'subqueries'**.
 - Alustuseks võib ülesannet lihtsustada ja teha päring ilma eelmist päringut arvestamata
4. Pärida andmebaasist jõe algus- ning lõpp-punkti koordinaadid ja leida nende punktide vaheline kaugus linnulennult. Tulemuse võib limiteerida näiteks 10 esimese kirje peale.
 - Kahe punkti vahelise kauguse leidmiseks kasutada järgnevat valemit:
$$d = R * \cos(\sin(lat1) * \sin(lat2) + \cos(lat1) * \cos(lat2) * \cos(lon1 - lon2));$$
 - *R* – maa raadius. Ühik vastavalt sellele, mis ühikutes soovid vastust väljastada.
 - Tabelitest päritavad väärtused on kraadides, kuid C math.h trigonomeetrilised funktsioonid vajavad argumente radiaanides.
 - *PQgetvalue* funktsioon tagastab string tüüpi tulemuse. Seega tuleks see ümber konverteerida ujupunktarvuks, et temaga oleks võimalik arvutusi teostada.

Abistavad materjalid

- Postgre dokumentatsioon - <http://www.postgresql.org/docs/9.1/interactive/index.html>
- Postgre libpq C teegi kasutamisest - <http://www.postgresql.org/docs/9.1/interactive/libpq.html>
- SQL lausete testimiseks võib kasutada mõnda klienti, nt <http://ewis.pld.ttu.ee/mywebsql/>
 - (kasutaja: student; parool: iag0582; andmebaas: ablid)
- Näidiskood <http://ati.ttu.ee/~hkinks/iag0582/praktikum11/psql.c>
 - gcc psql.c -lpq
- Kompileerimisel on soovitatav on kasutada ICT-502 arvuteid, kuna PostgreSQL teegid on sinna eelnevalt paigaldatud.
- Postgre libpq teegi lisamiseks
#include <psql/libpq-fe.h> + kompileerimisel käsk -lpq
- Math.h teegi lisamine:
#include <math.h> + kompileerimisel käsk -lm
- Põhilised libpq funktsioonid:

```
//Ühenduse loomine
PGconn *conn;
conn = PQconnectdb("dbname=<database name> host=<hostname> user=<username>
password=<password>");

//Kontrollime ühendust
if (PQstatus(conn) == CONNECTION_BAD) {
    //Kui ühendus ebaõnnestus
    /*Error handling*/
}

//päringu teostamine, nt pärimine 10 esimest kirjet tabelist <tabel>
PGresult *res;
res = PQexec(conn, "SELECT * FROM <table> LIMIT 10;");

//Kontrollime tulemust
if (PQresultStatus(res) != PGRES_TUPLES_OK) {
    //Kirjeid ei tagastatud
    /*Error handling*/
}

//kirjete arvu leidmine
int row_count = PQntuples(res);

//väärtuse pärimine vastavast reast ja veerust
char *val;
val = PQgetvalue(res, 0, 1); //tagastab väärtuse rida 0 veerg 1

//Vabasta PGresult handle
PQclear(res);

//Katkesta ühendus ja vabasta PGconn handle poolt hõivatud mälu
PQfinish(conn);
```