IN2090 – Obligatorisk Oppgave 5
Programmering med SQL

Publisert dato: 05.11.2020.

Innleveringsfrist: 19.11.2020 23:59.

Filer som skal leveres: 1: `administrator.py` eller `Administrator.java`.

Antall forsøk: 1.

Les gjennom hele teksten nøye før du begynner å løse oppgavene. Du kan velge om du ønsker å løse oppgaven med Python eller Java.

Oppgave 1 – Oppsett av databaser og programmer

I denne oppgaven skal du kun sette opp databasene som programmene du skriver i de neste oppgavene skal bruke. Du trenger ikke levere det du gjør i denne oppgaven.

Last ned Zip-filen[1] og unzip den (f.eks. med `unzip administrator.zip`). Kjør SQL-scriptet `webshop.sql` (fra den unzippede mappen) som setter opp databaseskjemaet slik:

```
psql -h dbpg-ifi-kurs01 -U brukernavn_priv -d brukernavn -f webshop.sql
```

hvor du erstatter `brukernavn` med ditt UiO-brukernavn. Deretter kjør SQL-scriptet `data.sql` som generere eksempel data for tabellene slik:

```
psql -h dbpg-ifi-kurs01 -U brukernavn_priv -d brukernavn -f data.sql
```

hvor du erstatter `brukernavn` med ditt UiO-brukernavn.

Databaseskjemaet `ws` inneholder nå følgende tabeller:

`ws.categories`(cid, name)

`ws.products`(pid, name, price, cid, description)

`ws.users`(uid, username, password, name, address)

`ws.orders`(oid, uid, pid, num, date, payed)

[1] `https://www.uio.no/studier/emner/matnat/ifi/IN2090/h20/obliger/administrator.zip`

hvor `ws.products(cid)` refererer til `ws.categories(cid)` og sier hvilken kategori et produkt tilhører, `ws.orders(uid)` refererer til `ws.users(uid)` og `ws.orders(pid)` refererer til `ws.products(pid)`. Altså, `ws.orders` inneholder bestillinger for en bruker gitt ved `uid` på et produkt gitt ved `pid`, hvor `num` er antall produkter bestilt, `date` er datoen for bestillingen, og `payed` har verdien `0` dersom bestillingen ikke ennå er betalt, og verdien `1` dersom bestillingen er betalt.

Merk: Scriptet `data.sql` generere tilfeldige bestillinger (i `ws.orders`-tabellen) for hver kjøring. Hver student for derfor en unik database, så dine resultater kan derfor være noe forkjellig fra dine medstudenters, samt eksempel-kjøringene under.

I mappen du laster ned er det et skelett-program (`python/administrator.py` for Python og `java/Administrator.java` for Java) som du skal fylle ut i oppgavene under. I programmet må du sette `dbname` lik ditt vanlige UiO-brukernavn, `user` lik brukeren din som slutter med `_priv`, og `pwd` lik det passordet du fikk for `_priv`-brukeren i mail fra USIT.

Python

Dersom du velger Python må programmene du skriver kjøres med Python 3. Dette gjøres på følgende måte:

```
python3 administrator.py
```

For Python kan det være (om du får en feilmelding om at `psycopg2` ikke finnes) du også må installere biblioteket `psycopg2`, som gjøres ved å kjøre:

```
pip3 install --user psycopg2-binary
```

Java

Dersom du velger å løse oppgaven med Java må ha med filen `postgresql.jar` i *classpath*'en når du kjører programmet. Filen ligger i samme mappe som `Administrator.java`. Du kjører så Java programmet (etter vanlig kompilering med `javac`) slik i Linux/Mac:

```
java -cp .:postgresql.jar Administrator
```

og slik i Windows:

```
java -cp ".;postgresql.jar" Administrator
```

Kjøring av programmer og fjerninnlogging

Programmene dere skriver skal kommunisere med databasen, og må derfor kjøres innenfor IFIs WiFi, på samme måte som ellers når dere logger inn i databasene. Altså må dere enten være fysisk på IFI, eller fjerninnlogge og kjøre programmene der. Dersom dere ønsker å både skrive og kjøre

programmene via fjerninnlogging kan dere kjøre følgende kommando fra IFI-maskinen (etter fjerninnlogging) for å laste ned filene direkte til ditt UiO-hjemmeområde:

```
wget https://www.uio.no/studier/emner/matnat/ifi/IN2090/h20/obliger/administrator.zip && \
unzip administrator.zip && \
cd administrator
```

Dersom du ønsker å skrive programmene lokalt på din maskin, og kun kjøre dem via fjerninnlogging, kan du flytte filene fra din maskin til UiO-hjemmeområdet ditt slik:

1. Først flytt alle filer du ønsker å flytte over i en egen mappe, f.eks. `move`
2. Zip-mappen med

   ```
   zip -r move.zip move
   ```

3. Flytt mappen til UiO med:

   ```
   scp move.zip <brukernavn>@login.ifi.uio.no:~/
   ```

4. Logg inn på IFI med vanlig SSH-kommando:

   ```
   ssh <brukernavn>@login.ifi.uio.no
   ```

5. Pakk ut mappen med:

   ```
   unzip move.zip
   ```

6. Hopp inn i mappen med:

   ```
   cd move
   ```

Oppgave 2 – Lage regninger

I denne oppgaven skal du implementere funksjonen `make_bills(conn)` (for Python) eller metoden `makeBills(Connection)` (for Java) som genererer regninger for brukere. Funksjonen skal spørre brukeren av programmet om et brukernavn. Dersom brukeren oppgir et brukernavn skal programmet så generere en regning for denne brukeren på følgende format:

```
Name: <navn>
Address: <adresse>
Total due: <total>
```

hvor `<navn>` er brukerens navn, `<adresse>` er brukerens adresse, og `<total>` er mengden penger brukeren skylder, altså summen av alle produktene som brukeren har bestilt men ennå ikke betalt for (husk at en bestilling kan inneholde flere av samme produkt via `num`-kolonnen).

F.eks. kan output da se slik ut (brukerens input er markert i rødt):

```
-- ADMINISTRATOR --
Please choose an option:
 1. Create bills
 2. Insert new product
 3. Exit
Option: 1
 -- BILLS --
Username: test

---Bill---
Name: Tester
Address: Teststreet 1
Total due: 25.0
```

Dersom brukeren ikke oppgir et brukernavn skal programmet generere en regning for alle brukere i databasen. F.eks. kan output da se slik ut (brukerens input er markert i rødt, merk ingen input for `Username:`):

```
-- ADMINISTRATOR --
Please choose an option:
 1. Create bills
 2. Insert new product
 3. Exit
Option: 1
 -- BILLS --
Username:

---Bill---
Name: Carl Smith
Address: Streetroad 34, 1234 Townplace
Total due: 404.88

---Bill---
Name: Mary Sagan
Address: Placestreet 12B, 4356 Nicetown
Total due: 259.91

---Bill---
Name: Tester
Address: Teststreet 1
Total due: 25.0
```

Oppgave 3 – Sette inn nye produkter

I denne oppgaven skal du implementere funksjonen `insert_product(conn)` (for Python) eller metoden `insertProducts(Connection)` (for Java) som lar brukeren sette inn nye produkter i Webshopens produktkatalog (som da

er `ws.products`-tabellen). Brukeren skal bli spurt om navn og pris på produktet, navnet på kategorien produktet tilhører, samt en mulig beskrivelse.

Programmet skal så sette dette produktet inn i `ws.products`-tabellen. En kjøring kan da se slik ut (brukerens input er markert i rødt):

```
-- ADMINISTRATOR --
Please choose an option:
 1. Create bills
 2. Insert new product
 3. Exit
Option: 2

 -- INSERT NEW PRODUCT --
Product name: Juice
Price: 2.3
Category: food
Description: Fresh orange juice
New product Juice inserted.
```

Etter dette skal det finnes et nytt tuppel i tabellen `ws.products` med `name` lik `Juice`, `price` lik `2.3` og `description` lik `Fresh orange juice`.

Merk at `pid`-kolonnen i `ws.products` er satt til `SERIAL`, så verdien for denne genereres automatisk. Merk også at brukeren oppgir navnet på kategorien, men i `ws.products`-tabellen skal vi ha `cid`, så man må altså finne `cid`-verdien basert på kategori-navnet fra brukeren gjennom en `SELECT`-spørring (husk at dette kan gjøres ved å bruke `SELECT` i en `INSERT`-spørring).

Levering

Obligatoriske oppgaver skal leveres i Devilry[2]. Sørg for at du er registrert i systemet ved å logge inn og se at `oblig5` er tilgjengelig som en oblig for IN2090. *Sjekk dette før du begynner å løse oppgavene!* Dersom du ikke er registrert, send en mail til `leifhka@ifi.uio.no`.

Du kan levere så mange ganger du vil, det er kun den siste leveringen som teller. Alle leveringer som blir lastet opp etter fristen vil ikke bli godkjent med mindre man har en godkjent grunn for å levere sent, se mer informasjon om dette på IFIs nettsider[3].

Lykke til!

[2] `https://devilry.ifi.uio.no/`

[3] `https://www.uio.no/studier/admin/obligatoriske-aktiviteter/mn-ifi-oblig.html`