DAT107 Databaser – våren 2023

Obligatorisk øvelse 3: JPA

Lars-Petter Helland, 22.03.2023

Rammer

Oppgaven løses i grupper på inntil 4 deltagere. Du kan også levere alene.

NB! Oppgaven skal leveres via en av gruppene i gruppesettet 'Oblig 3 Gr'. Dette gjelder også om du leverer alene. (Finn et ledig gruppenummer og legg dere inn selv.) I**nnleveringer utenfor dette gruppesettet risikerer å ikke bli rettet!!**

Vurdering og godkjenning

Skriftlig innlevering i Canvas innen **torsdag 12. april kl.23:59**.

Databasen som brukes i prosjektet må være på **ider-database.westeurope.cloudapp.azure.com** slik at innlevert prosjekt kan kjøres fra hvilken som helst PC (f.eks. læreren sin PC ved retting). **Ved innlevering må databasen være opprettet og inneholde de nødvendige data. Java-prosjektet må inneholde riktig oppkoblingsinfo, dvs. datavasenavn, brukernavn og passord**. NB! Ikke bruk samme passord her som du bruker andre steder.

Leveranse i Canvas skal være én zip-fil med følgende innhold:

- Et pdf-dokument med kopier av skjermbilder som viser kjøring av applikasjonen. **NB! I starten av dette dokumentet skal det stå gruppenummer og navn og studentnummer til alle som er med på innleveringen.**
- Eclipse-prosjektet, inkl. SQL-skriptet som er brukt for å opprette databasen

---

Problemstilling

Det skal lages et lite Java-program som holder orden på ansatte og prosjekter i et firma. Data som skal lagres i databasen er:

Ansatt

- Unik ansatt-id (automatisk generert løpenummer)
- Unikt brukernavn (initialer, 3-4 bokstaver, f.eks. «lph»)
- Fornavn
- Etternavn
- Dato for ansettelse
- Stilling
- Månedslønn
- Hvilken avdeling den ansatte jobber i
- Hvilke prosjekter den ansatte deltar/har deltatt i m/ rolle og antall arbeidstimer

Avdeling

- Unik avdeling-id (automatisk generert løpenummer)
- Navn
- Hvilken ansatt som er sjef

Prosjekt

- Unik prosjekt-id (automatisk generert løpenummer)
- Navn
- Beskrivelse
- Hvilke ansatte som deltar har deltatt m/ rolle og antall arbeidstimer

Assosiasjoner (forhold) og integritetsregler (skranker) for dataene

- En ansatt **må** jobbe i en avdeling
- En avdeling kan ha flere ansatte, men **må** ha minst én ansatt pga. krav om sjef (se under)
- En avdeling **må** ha en sjef (en ansatt, som også jobber i denne avdelingen)
- Det skal ikke være mulig å slette en avdeling hvis det er ansatte som jobber der
- Det skal ikke være mulig å slette en ansatt hvis vedkommende er sjef i en avdeling
- En ansatt kan jobbe i flere prosjekter, og har en rolle og et timetall for hvert prosjekt
- Et prosjekt kan ha flere ansatte som jobber/har jobbet der, og har rolle og timetall for hver ansatt
- Det skal ikke være mulig å slette en ansatt hvis vedkommende har registrert timer i et prosjekt
- Det skal ikke være mulig å slette et prosjekt hvis det er ansatte tilknyttet prosjektet

Hva skal programmet gjøre

Det skal lages et helt enkelt interaktivt Java-program som gjør diverse databaseoperasjoner. Detaljene står utdypet lenger nede. Det er ikke nødvendig å lage et "fancy" program. Gjør det enkelt.

Forslag til fremgangsmåte

Når man får en litt større oppgave, er det viktig å få delt opp arbeidet på en fornuftig måte. Et par mulige fremgangsmåter kan være:

- Først komplett datamodellering og oppretting av databasen, deretter programmering av JPA-tingene, og til slutt programmering av det menystyrte programmet og testing at det virker. (Waterfall)

alternativt:

- Først en komplett miniutgave av datamodell + database + JPA + program inkl. testing, deretter litt mer av alt, deretter litt mer av alt, ..., helt til vi er ferdige. (Iterativ og inkrementell)

Jeg vil absolutt anbefale den siste (iterative) fremgangsmåten. Hvorfor? Det har i hovedsak med feedback, læring underveis og risikostyring å gjøre.

I den første fremgangsmåten (Waterfall) vil vi oppdage feil sent, og vi legger ikke opp til å kunne forbedre det vi har gjort basert på læring.

I den andre fremgangsmåten (Iterativ og inkrementell) får vi kontroll på alle de kompliserte tingene ganske tidlig, og på slutten er det i hovedsak å utvide og forbedre det man har.

Det er likevel ikke noe i veien å lage en datamodell først for å få oversikt over det man skal frem til. Men prøver man å implementere hele datamodellen fra starten stiller dette større krav til Java-koden for å få ting til å virke slik at man får testet.

Resten av teksten er delt opp etter foreslåtte iterasjoner.

Iterasjon1 - Få kontakt med databasen fra et Java-program

Som sagt er det sikkert lurt å jobbe **iterativt**, dvs. at dere løser en mindre del av problemstillingen «ferdig» før dere går videre.

Første iterasjon har som mål å få alle deler av applikasjonen (database, JPA, app) til å virke sammen. Dette er et viktig mål, og derfor noe som må prioriteres.

Jeg foreslår at dere begynner med **Ansatt**, og ser bort fra Avdeling og Prosjekt. Den ansatte har da altså **unik ansatt-id**, **unikt brukernavn, fornavn**, **etternavn**, **dato for ansettelse**, **stilling** og **månedslønn**.

Forslag til fremgangsmåte:

- Opprett et Eclipse-prosjekt og legg inn jar-filer og (tom) META-INF/persistence.xml
- Definér databasen (kun ansatt i denne omgang) inkl. litt eksempeldata, dvs. lag et SQL-skript for dette og kjør i PgAdmin/DBeaver/Eclipse. Sjekk innholdet i tabellen via PgAdmin/DBeaver/Eclipse.
- Lag entitetstypen Ansatt (i Java) med nødvendige annoteringer, gettere/settere og skrivUt().
- Gjør nødvendige tilpassinger i persistence.xml
- Lag et lite Main-program for å teste at alt er satt opp riktig. Hent f.eks. ut en ansatt og skriv ut på skjermen. (EntityManagerFactory, EntityManager, try, find(), finally, close).

√ Nå har dere allerede kommet ganske langt ☺

Iterasjon2 - Plan for god programstruktur og menystyrt program

Det neste å tenke på (hvis dere ikke allerede har gjort det) er å lage en hjelpeklasse for databasetilgangen. Kall denne AnsattDAO (DAO betyr Data Access Object) og legg inn metoden **`public Ansatt finnAnsattMedId(int id)`**. Skriv om Main-programmet slik at du bruker denne hjelpeklassen til å hente en ansatt fra databasen.

Det vi skal ende opp med til slutt er et enkelt interaktivt program som holder orden på ansatte og prosjekter osv. …

Hvordan dere løser selve brukergrensesnittet er litt opp til dere, men det er kanskje naturlig med et menysystem med brukerdialog for de enkelte menypunktene.

Menyvalg dere kan lage i første omgang er:

- Søke etter ansatt på ansatt-id
- Søke etter ansatt på brukernavn (initialer)
- Utlisting av alle ansatte
- Oppdatere en ansatt sin stilling og/eller lønn
- Legge inn en ny ansatt

Husk å lage metoder i AnsattDAO for hver av tingene dere ønsker å gjøre i programmet som har med databasen å gjøre. Prøv å lag ryddig kode.

(Hvis det blir mye inntasting i brukergrensesnittet (f.eks. ved registrering av ny ansatt) kan dere bruke til dels hardkodete data for å slippe inntasting. Det viktigste er databasegreiene.)

√ Bra jobbet. Resten burde jo være plankekjøring nå? ☺

Iterasjon3 - Utvide databasen med Avdeling

Det neste kan da bli å lage Avdeling, og å koble sammen Ansatt og Avdeling. Det er to ulike forhold (relationship) mellom ansatte og avdeling:

1. En ansatt må jobbe i en avdeling
2. En avdeling må ha én sjef (som jobber i avdelingen)

Forslag til fremgangsmåte:

- Utvid databasedefinisjonen til nå å inneholde både Ansatt og Avdeling, og sett opp koblingene mellom disse. Legg inn eksempeldata for f.eks. 10 ansatte og 3 avdelinger.

  Høna-og-egget-tips: Siden en ansatt må jobbe i en avdeling, og en avdeling må ha en sjef, er det vanskelig å legge inn data for den ene før den andre. Trikset her er å f.eks. vente med Avdeling sin sjef, og legge dette inn via en UPDATE etter at den ansatte er opprettet. Problemstillingen gjelder også tabelldefinisjonene. Her må man gjøre ALTER TABLE ... ADD CONSTRAINT på en av fremmednøklene etter at begge tabeller er opprettet.

- Sjekk innholdet i tabellene via PgAdmin/DBeaver/Eclipse.
- Lag entitetstypen Avdeling (i Java) og oppdater entitetstypen Ansatt (i Java) slik at forholdene mellom dem er satt opp.
- Legg til Avdeling i persistence.xml
- Lag en hjelpeklasse AvdelingDAO og legg inn metoden public Avdeling finnAvdelingMedId(int id).
- Sjekk via et main-program at denne virker.

√ Enda et hinder er unnagjort ☺

Nå begynner det kanskje å bli vanskelig, og dere har allerede brukt mye tid på øvingen. Jeg tror det er god læring i å gjøre hele Iterasjon4 og Iterasjon5, men det er lov å levere uten å ha gjort disse 100% ferdig. Iterasjon4 inneholder en del tricky logikk, mens Iterasjon5 er god trening i mange-til-mange!

Og som tidligere sagt: Ikke bruk for mye energi på brukergrensesnittet. Det viktigste er å få øvd på databasetingene og JPA.

Iterasjon4 - Utvide programmet med flere menypunkter

Nå kommer jeg ikke til å si mer i detalj hvilke hjelpemetoder AnsattDAO og AvdelingDAO bør utvides med for å løse problemstillingene.

Utvid det interaktive programmet med følgende funksjoner:

- Søke etter ansatt på ansatt-id
- Søke etter ansatt på brukernavn (initialer)
- Utlisting av alle ansatte
- Oppdatere en ansatt sin stilling og/eller lønn
- Endring: Legge inn en ny ansatt må nå også angi hvilken avdeling vedkommende skal jobbe på

- Søke etter avdeling på avdeling-id
- Utlisting av alle ansatte på en avdeling inkl. utheving av hvem som er sjef
- Oppdatere hvilken avdeling en ansatt jobber på. Man kan ikke bytte avdeling hvis man er sjef!
- Legge inn en ny avdeling. NB! Siden en avdeling MÅ ha en sjef, må man velge blant en av de allerede eksisterende ansatte (som da jobber i en annen avdeling). Den ansatte som blir sjef i den nye avdelingen skal automatisk overføres til avdelingen vedkommende blir sjef for.

√ Uffameg. Dette var tricky. Men det føles vel godt å ha funnet ut av det. ☺

Iterasjon5 - Få på plass Prosjekter og Prosjektdeltagelse

Det siste er dette med Ansatt, Prosjekt og Prosjektdeltagelse.

Jeg overlater til dere å modellere og implementere dette i SQL.

Igjen tror jeg en grei fremgangsmåte begynner med å legge inn noe data via SQL, og ta noen utskrifter fra Java for å sjekke at ting ser greit ut.

Utvid det interaktive programmet med følgende funksjoner:

- Søke etter ansatt på ansatt-id
- Søke etter ansatt på brukernavn (initialer)
- Utlisting av alle ansatte
- Oppdatere en ansatt sin stilling og/eller lønn
- Endring fra sist: Legge inn en ny ansatt må nå også angi hvilken avdeling vedkommende skal jobbe på
- Søke etter avdeling på avdeling-id
- Utlisting av alle ansatte på en avdeling inkl. utheving av hvem som er sjef
- Oppdatere hvilken avdeling en ansatt jobber på. Man kan ikke bytte avdeling hvis man er sjef!
- Legge inn en ny avdeling. NB! Siden en avdeling MÅ ha en sjef, må man velge blant en av de allerede eksisterende ansatte (som da jobber i en annen avdeling). Den ansatte som blir sjef i den nye avdelingen skal automatisk overføres til avdelingen vedkommende blir sjef for.

- Legge inn et nytt prosjekt
- Registrere prosjektdeltagelse (ansatt med rolle i prosjekt)
- Føre timer for en ansatt på et prosjekt
- Utskrift av info om prosjekt, inkl. liste av deltagere med rolle og timer, og totalt timetall for prosjektet

√ Supert. Det var det. ☺

Håper denne øvelsen hjalp på å lære både praktisk modellering, SQL og JPA …

Lars-Petter, 22. mars 2023