Eksamensoppgave i Databasesystemer

Forfattere: Ida Tollaksen (233094) & Emil Berglund (233099)

Gruppe 40 – Canvas

Høgskolen i Østfold – 15.03.2024

Innholdsfortegnelse

[Kravspesifikasjon: 2](#_Toc160981812)

[Normalisering 4](#_Toc160981813)

[ER modellering: 5](#_Toc160981814)

[Dummy data 6](#_Toc160981815)

[SQL spørringer 7](#_Toc160981816)

[Kilder: 10](#_Toc160981817)

# Kravspesifikasjon:

Brukere – Brukeradministrasjon

* Databasesystemet skal ha brukere med unik ID i tillegg til relevant informasjon som «fornavn», «etternavn», «alder» og kjønn.
* Brukerne skal differensieres som enten "utøver" eller "trener".

Økt – Overblikk over kommende/tidligere økter

* Tabellen «økt» skal si hvordan type økt en utøver skal gjøre, og da også per økt.
* Hver økt får sin egen ID, med «økttype», «dato» i tillegg til «brukerid» som sier noe om hvilke bruker som skal gjøre hvilke økt til hvilken dato.
* En økt kan være en kondisjonsøkt, en styrkeøkt eller en konkurranse.

Øvelser – Oversikt over øvelser

* Tabellen «øvelser» skal inneholde alle øvelsene en utøver kan gjøre, eller som en trener kan tildele.
* Hver øvelse får tildelt en unik ID, og har et navn og en beskrivelse.

Øvelselogg – Logg av treningsøkt (treningsøvelser)

* Tabellen «Øvelselogg» skal være logger over alle øvelser en utøver gjør i løpet av en økt.
* Hver logg får tildelt en unik ID, og inkluderer brukerens ID, øvelsens ID, dato og puls-sone.
* Avhengig av typen øvelse, kan det også legges til informasjon om repetisjoner og vekt, eller distanse.

Programmer – Oversikt over opprettede programmer

* Tabellen «Programmer» skal vise til opprettede programmer.
* Hvert program får tildelt en unik ID og et beskrivende navn, som kan referere til ukedager eller annet.
* Tabellen viser også oppvarming og avslutning for hvert program.

Program øvelser – Øvelser tilhørende programmer

* Tabellen "Program\_øvelser" er en oppslagstabell som knytter øvelser til programmer.
* Tabellen skal inneholde en «program\_id» som skal matche «program\_id» i tabellen «programmer». I tillegg skal det også være «ØvelseID» som sier noe om hvilke øvelser som skal gjøres.

Ukeoversikt – Oversikt over hvilken plan utøvere følger

* Tabellen “Ukeoversikt” skal inneholde informasjon om hvilket program en spesifikk utøver følger, navnet på treneren som tildelte/opprettet denne planen, i tillegg til start og sluttdato for programmet.
* Hver ukeoversikt skal ha sin egen unike ID.

Ukeplan – Henvisning til program

* Tabellen «ukeplan» skal være en oppslagstabell som kan henvises til for å vise hvilke program som skal brukes.
* Den skal ha en primærnøkkel «PlanID» og en foreign key «ProgramID» som refererer til programmer.

Ukeplan\_ukedager – Henvisning til hvilke plan som skal brukes til ulike datoer

* Tabellen «ukeplan\_ukedager» skal være en oppslagstabell for å se hvilke plan man skal bruke i henhold til datoen man skal trene.
* Tabellen skal ha en PlanID som referer til «ukeplan»
* Tabellen skal ha en «ukedag» som viser til datoen hvor planene skal brukes.

# Normalisering

For å normalisere modellen vår til normalform BCNF startet vi med å samle all data i en stort excel tabell.

1NF – Første Normalform

For å få modellen vår til 1NF startet vi med å sørge for at alle verdiene våre var atomære, altså at vi ikke hadde lister i koloner og at all data i en kollene var av samme datatype. Deretter undersøkte vi hvilken data som blir redundant og dermed tar opp unødvendig lagringsplass i databasen. Vi evaluerte så hvordan man kan opprette separate tabeller for å lagre informasjonen mer effektivt.

2NF – Andre Normalform

Når vi jobbet med å få database modellen vår til andre normalform, måtte vi fjerne eventuelle partielle avhengigheter. En partiell avhengighet oppstår når en kolonne er avhengig av en del av primærnøkkelen, men ikke hele nøkkelen. Dette fører ofte til et rotete forhold mellom kolonnene innad i tabellen i tillegg til at det skaper en unødvendig rotete struktur.

3NF – Tredje Normalform

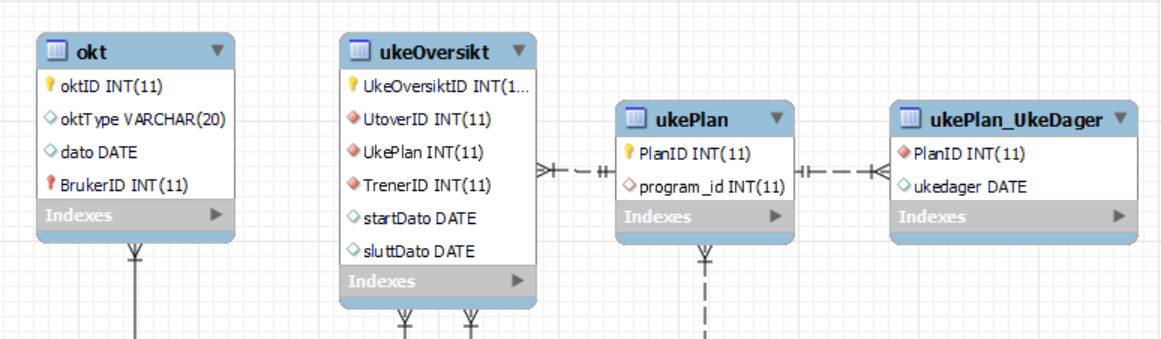
For å få modellen fra andre til tredje normalform måtte vi fjerne alle transitive avhengigheter. Transitive avhengigheter oppstår når to kolonner i en tabell er gjensidig avhengige av hverandre og i tillegg avhenger av primærnøkkelen. Når en tabell inneholder transitive avhengigheter, inneholder den mest sannsynlig også redundante data, noe som tar unødvendig lagringsplass i databasen.

BCNF – Boyce-Codd Normalform

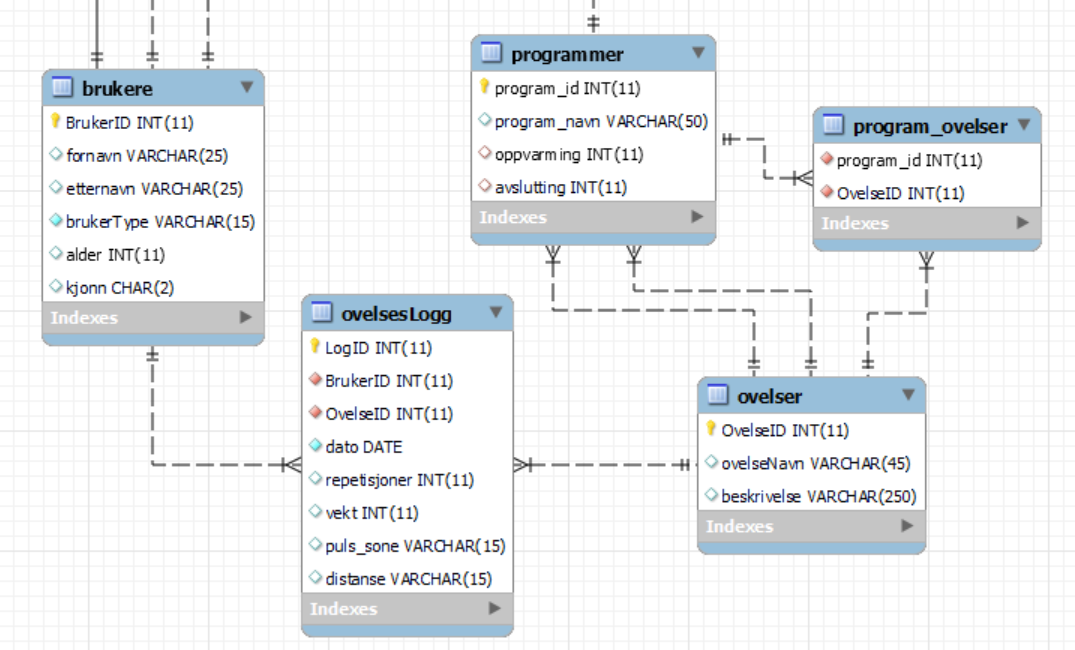
BCNF, eller Boyce-Codd normalform, er en normaliseringsform som bygger på 3NF. Denne normalformen er enda mer restriktiv enn tredje normalform og sikter mot å oppnå en enda bedre database struktur. For å få modellen vår til BCNF måtte vi sørge for at alle minimale determinanter var kandidatnøkler. Dette gjorde vi ved å sikre at alle våre funksjonelle avhengigheter var en triviell funksjonell avhengighet.

Ved å oppnå BCNF på databasen vår kunne vi sikkert si at alle funksjonelle avhengigheter i databasen var godt definerte og at vi ikke hadde noe redundant data. Dette gir oss en veldig god databasestruktur som er lett å vedlikeholde over tid.

# ER modellering:



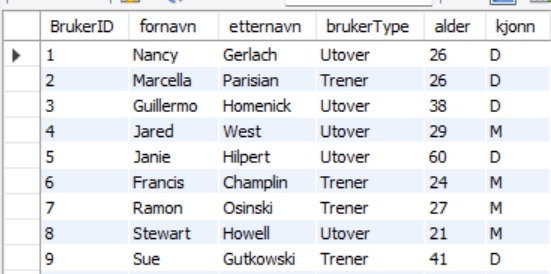




# Dummy data

For å fylle tabellene våre skrev vi insert into script og vi fikk deretter ChatGPT til å generere opp i større mengder. Se kilde 1 for ChatGPT spørringen[[1]](#footnote-2).

Dummy data i “brukere”-tabellen vår:



**INSERT** **INTO** brukere (brukerID, fornavn, etternavn, brukerType, alder, kjonn) **VALUES**  
(1, 'Nancy', 'Gerlach', 'Kunde', 26, 'D'),  
(2, 'Marcella', 'Parisian', 'Trener', 26, 'D'),  
(3, 'Guillermo', 'Homenick', 'Kunde', 38, 'D'),  
(4, 'Jared', 'West', 'Kunde', 29, 'M'),  
(5, 'Janie', 'Hilpert', 'Kunde', 60, 'D'),  
(6, 'Francis', 'Champlin', 'Trener', 24, 'M'),  
(7, 'Ramon', 'Osinski', 'Trener', 27, 'M'),  
(8, 'Stewart', 'Howell', 'Kunde', 21, 'M'),  
(9, 'Sue', 'Gutkowski', 'Trener', 41, 'D'),

Vi satte totalt in 500 rader med dummy data for brukere tabellen.

Det samme ble gjort for de andre tabellene også, men for å unngå gjentakelser, viser vi kun ett eksempel.

# 

# SQL spørringer

1. Views:

Bruker\_oversikt\_uke:

**CREATE** VIEW bruker\_oversikt\_uke **AS**

**SELECT**

ukeOversikt.UtoverID **AS** BrukerID,

**CONCAT**(brukere.fornavn, ' ', brukere.etternavn) **AS** UtøverNavn,

ukeOversikt.TrenerID **AS** TrenerID,

**CONCAT**(trener.fornavn, ' ', trener.etternavn) **AS** TrenerNavn,

ukePlan\_UkeDager.ukedager **AS** Ukedag,

ukeOversikt.UkePlan **AS** PlanID,

programmer.program\_navn **AS** ProgramNavn,

okt.oktID,

okt.oktType

**FROM**

ukeOversikt

**JOIN**

brukere **ON** ukeOversikt.UtoverID = brukere.BrukerID

**JOIN**

brukere **AS** trener **ON** ukeOversikt.TrenerID = trener.BrukerID

**JOIN**

ukePlan **ON** ukeOversikt.UkePlan = ukePlan.PlanID

**JOIN**

ukePlan\_UkeDager **ON** ukePlan.PlanID = ukePlan\_UkeDager.PlanID

**JOIN**

programmer **ON** ukePlan.program\_id = programmer.program\_id

**JOIN**

okt **ON** ukePlan\_UkeDager.ukedager = okt.dato;

Bruker\_progresjon:

**CREATE** VIEW bruker\_progresjon **AS**

**SELECT**

ovelsesLogg.BrukerID,

brukere.fornavn,

brukere.etternavn,

ovelsesLogg.OvelseID,

ovelser.ovelseNavn,

**MIN**(ovelsesLogg.dato) **AS** forsteDato,

**MAX**(ovelsesLogg.dato) **AS** sisteDato,

**MIN**(ovelsesLogg.vekt) **AS** minVekt,

**MAX**(ovelsesLogg.vekt) **AS** maxVekt,

**MIN**(ovelsesLogg.repetisjoner) **AS** minRepetisjoner,

**MAX**(ovelsesLogg.repetisjoner) **AS** maxRepetisjoner

**FROM**

ovelsesLogg

**JOIN**

brukere **ON** ovelsesLogg.BrukerID = brukere.BrukerID

**JOIN**

ovelser **ON** ovelsesLogg.OvelseID = ovelser.OvelseID

**WHERE**

ovelsesLogg.OvelseID **NOT** **IN** (**1**, **3**)

**GROUP** **BY**

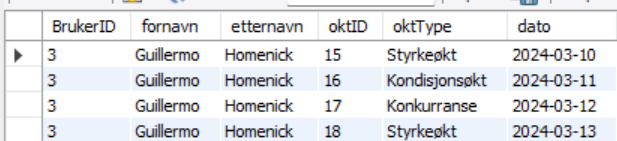
ovelsesLogg.BrukerID, ovelsesLogg.OvelseID;

1. Spørringer

Henter informasjon fra brukere og økt tabellen om brukeren som har bruker id 3.

**SELECT** b.BrukerID, b.fornavn, b.etternavn, o.oktID, o.oktType, o.dato  
**FROM** brukere **AS** b  
**JOIN** okt **AS** o **ON** b.BrukerID = o.BrukerID  
**WHERE** o.BrukerID = '3';

Resultat:

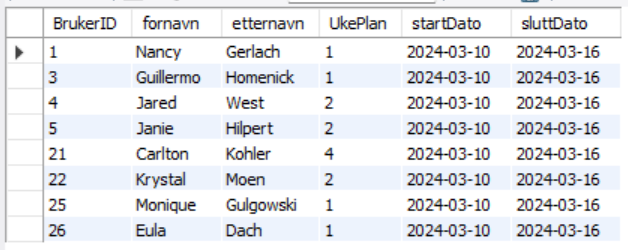


Henter ut informasjon om brukere som utfører en ukeplan som har start dato “2024-03-10” og slutt dato “2024-03-16”.

Spørring:

**SELECT** b.BrukerID, b.fornavn, b.etternavn, u.UkePlan, startDato, sluttDato  
**FROM** brukere **AS** b  
**JOIN** ukeOversikt **AS** u **ON** b.BrukerID = u.UtoverID  
**WHERE** startDato = '2024-03-10' **and** sluttDato = '2024-03-16';

Resultat:



Spørring:

**SELECT**

(**SELECT** **AVG**(alder) **FROM** brukere **WHERE** brukerType = 'Trener') **AS** gjennomsnittsalderUtover,

(**SELECT** **AVG**(alder) **FROM** brukere **WHERE** brukerType = 'Utover') **AS** gjennomsnittalderTrener;

Resultat:

Et bilde som inneholder tekst, Font, line, hvit

Automatisk generert beskrivelse

Henter ut gjennomsnittsalder for både trenere og utøvere i to separate kolonner.

Spørring:

**SELECT** OvelseID, ovelseNavn

**FROM** ovelser

**WHERE** OvelseID **NOT** **IN** (**SELECT** **DISTINCT** OvelseID **FROM** ovelsesLogg);

Resultat:

Et bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelse

Henter ut alle øvelser med ID som ikke har blitt skrevet logg på.

# Kilder:

**Kilde 1 – ChatGPT spørring for dummy data**

ChatGPT 3.5. (2024). Openai.com. <https://chat.openai.com/>

**‌**

ChatGPT spørring:

*Basert på dette, generer opp mer tilfeldig dummy data til tabellen, verdiene under kjøn skal enten være “D” eller “M”: INSERT INTO bruker (brukerID, fornavn, etternavn, brukerType, alder, kjonn) VALUES (1, 'Nancy', 'Gerlach', 'Kunde', 26, 'D')*

Svar:

Here's an SQL INSERT INTO script for the "bruker" table:

**INSERT** **INTO** bruker (brukerID, fornavn, etternavn, brukerType, alder, kjonn) **VALUES**  
(1, 'Nancy', 'Gerlach', 'Kunde', 26, 'D'),  
(2, 'Marcella', 'Parisian', 'Trener', 26, 'D'),  
(3, 'Guillermo', 'Homenick', 'Kunde', 38, 'D'),  
(4, 'Jared', 'West', 'Kunde', 29, 'M'),  
(5, 'Janie', 'Hilpert', 'Kunde', 60, 'D'),  
(6, 'Francis', 'Champlin', 'Trener', 24, 'M'),  
(7, 'Ramon', 'Osinski', 'Trener', 27, 'M'),  
(8, 'Stewart', 'Howell', 'Kunde', 21, 'M'),  
(9, 'Sue', 'Gutkowski', 'Trener', 41, 'D'),  
(10, 'Oscar', 'Kub', 'Trener', 37, 'M'),  
(11, 'Frankie', 'Kilback', 'Trener', 56, 'D'),  
(12, 'Jared', 'Walter', 'Trener', 34, 'D')

This script will insert all the provided data into the "bruker" table, assuming that the columns are correctly set up in your database.

**Kilde 2 - hilite.me**

Source code beautifier / syntax highlighter – convert code snippets to HTML «hilite.me. (2024). <http://hilite.me/>

1. [↑](#footnote-ref-2)