

# Database rapport

## 02327 Indledende databaser og database programmering

Projektnavn: Database Rapport

Gruppe nr: 11

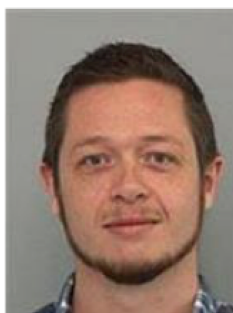
Afleveringsfrist: 13/05 - 2016

Denne rapport er afleveret via CampusNet (der skrives ikke under).

Denne rapport indeholder 34 sider inkl. denne side.



**S136396**  
**Christensen, Brian**



**S136378**  
**Christiansen, Morten Due**



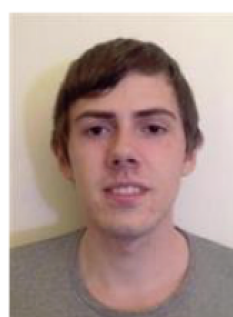
**S140330**  
**Danielsen, Casper**



**S153670**  
**Larsen, Sofie Paludan**



**S153869**  
**Lindberg, Cecilie**



**S123620**  
**Matthiessen, Niels Ulrich**

	Analyse	Design	Impl.	Test	Dok.	Ialt
<b>Morten</b>	7	4	9	6	4	30
<b>Brian</b>	8	4	8	5	5	30
<b>Casper</b>	9	3	8	6	4	30
<b>Sofie</b>	8	3	7	8	4	30
<b>Cecilie</b>	8	3	9	6	4	30
<b>Niels</b>	7	5	8	5	5	30

## Indhold

<b>1</b>	<b>Indledning</b>	<b>4</b>
<b>2</b>	<b>Resumé</b>	<b>4</b>
<b>3</b>	<b>Udleveret Database</b>	<b>4</b>
3.1	Brugerroller . . . . .	5
3.2	Produktbatch dato . . . . .	5
<b>4</b>	<b>ER diagram</b>	<b>5</b>
<b>5</b>	<b>Relationsdiagram</b>	<b>6</b>
<b>6</b>	<b>Normalisering</b>	<b>7</b>
<b>7</b>	<b>Brugere og rettigheder</b>	<b>8</b>
7.1	Rollebeskrivelse . . . . .	8
7.2	Rettigheder . . . . .	8
<b>8</b>	<b>Views</b>	<b>9</b>
<b>9</b>	<b>Stored Procedures</b>	<b>10</b>
<b>10</b>	<b>Triggers</b>	<b>12</b>
<b>11</b>	<b>Transaktionslogik</b>	<b>12</b>
<b>12</b>	<b>Applikations Interface</b>	<b>13</b>
<b>13</b>	<b>Test</b>	<b>14</b>
<b>14</b>	<b>Opgaver</b>	<b>16</b>
14.1	Query 1 . . . . .	16
14.2	Query 2 . . . . .	16
14.3	Query 3 . . . . .	17
14.4	Query 4 . . . . .	18
14.5	Query 5 . . . . .	19
14.6	Query 6 . . . . .	20
14.7	Query 7 . . . . .	21
14.8	Query 8 . . . . .	21
14.9	Query 9 . . . . .	22
<b>15</b>	<b>Konklusion</b>	<b>23</b>

<b>16</b>	<b>Appendiks: Konfiguration</b>	<b>24</b>
16.1	Klon Repository . . . . .	24
16.2	PC konfiguration . . . . .	24
<b>17</b>	<b>Bilag</b>	<b>26</b>
17.1	SQL kode . . . . .	26
<b>18</b>	<b>Litteraturliste</b>	<b>33</b>

## 1 Indledning

I dette projekt skal der implementeres et databaseinterface i Java, der gør det muligt at kommunikere med vores database med queries. Det skal også være muligt at hente og oprette data fra Java, der også ændres i databasen.

## 2 Resumé

Opgaven i dette projekt har været, at reverse engineer en database, der er blevet udleveret. Udover dette skulle der implementeres views, stored procedures og lignende. Der skulle også udvikles et Java interface til at kommunikere med databasen.

Tabellerne i den udleverede database er analyseret og konfigureret. Der er tilføjet en rolletabel, og der er fjernet initialer fra operatoer tabellen.

ER-diagrammer og relationsdiagrammer er udformet, og der er kontrolleret for normalisering i første, anden og tredje normalform.

Der er analyseret brugerrettigheder, samt oprettet en databasebruger med begrænset adgang til tabeller.

Der er oprettet en række views til udtræk af data på tværs af tabellerne, og der er oprettet stored procedures til sikker indsættelse af data i databasen.

Triggers er blevet overvejet til løsningen, men ikke brugt. Der er blevet lavet transactions logic til sikker ændring af flere tabeller i databasen.

Der er blevet oprettet et Applikationsinterface i Java, som kan kommunikere med databasen, og der er lavet grundig test af hele implementationen.

Opgaven indeholder også en række eksempler på sql scripts brugt og komplet sql kode kan ses i bilag.

## 3 Udleveret Database

Der er blevet udleveret en .sql-fil med et databaseskema. Denne indeholder følgende tabeller:

operatoer(opr\_id : integer; opr\_navn : string; ini : string; cpr : string; password : string)

raavare(raavare\_id : integer; raavare navn : string; leverandoer : string)

raavarebatch(rb\_id : integer; raavare\_id : integer; maengde : real)

recept(recept\_id : integer; recept\_navn : string)

receptkomponent(recept\_id : integer; raavare\_id : integer; nom\_netto : real; tolerance : real)

produktbatch(pb\_id : integer; recept\_id : integer; status : integer)

produktbatchkomponent(pb\_id : integer; rb\_id : integer; opr\_id : integer;  
tara : real; netto : real)

Ud fra analyse af CDIO Final oplægget [CDI16] er det blevet klarlagt, at ini attributten i operatoertabellen ikke bliver brugt, da vi ikke kan se nogen brugbar betydning af denne. Denne er derfor blevet slettet fra tabellen.

### 3.1 Brugerroller

CDIO Final oplægget kræver, at der er brugerroller. Rollerne skal fungere på sådan en måde, at den laveste rolle kun har adgang til en bestemt række ting, mens rollen højere skal have adgang til sit eget, samt det i rollen under osv.

Grundet dette er der blevet oprettet en tabel "roller" med attributterne `rolle_id` og `rolle_navn`. Samtidig har operatoertabellen fået en attribut kaldet `rolle_id`, som er en fremednøgle, der refererer til denne relation. Her ved er der oprettet en "en til mange" relation, da flere operatører kan have samme rolle.

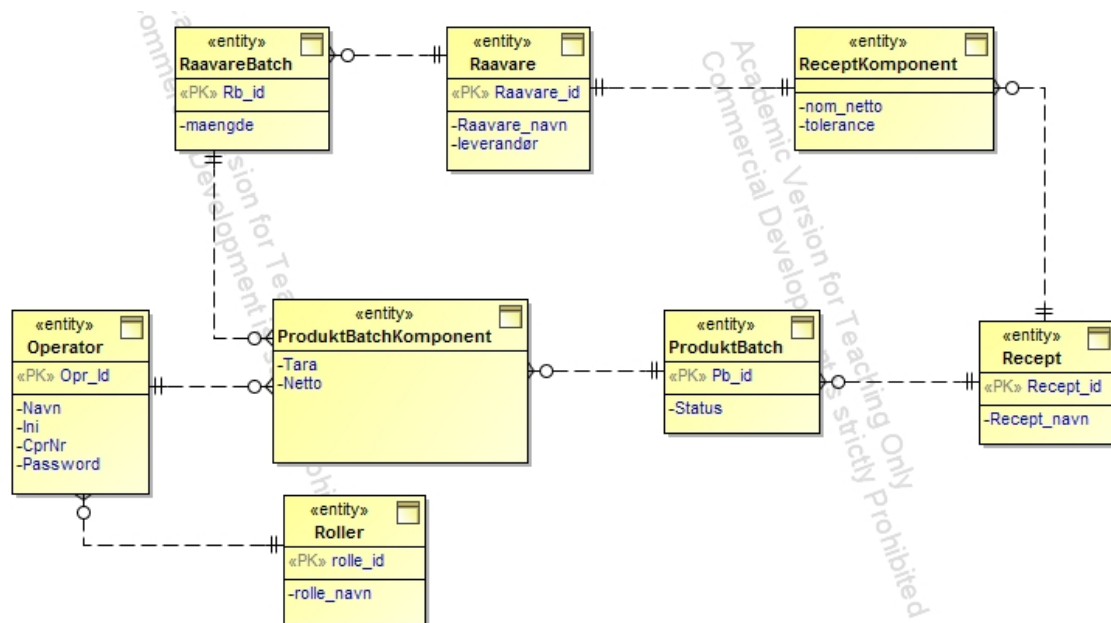
### 3.2 Produktbatch dato

CDIO Final oplægget kræver også, at Produktbatch relationen får en dato for oprettelse. Der er derfor tilføjet en attribut med navnet "dato", som er af typen Date. Når der bliver oprettet et nyt produktbatch, bliver datoen automatisk sat til datoen for den dag, den er blevet lavet.

## 4 ER diagram

ER-diagrammet bruges til at se forholdet mellem entities i databasen.

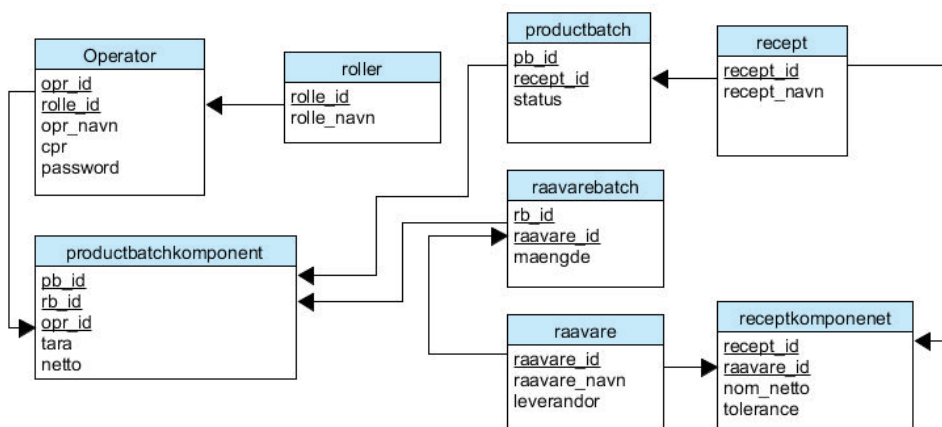
Ud fra diagrammet kan man for eksempel se, at en Recept kan bruges til flere ProduktBatch.



Figur 4.1: ER diagram

## 5 Relationsdiagram

I relationsdiagrammet kan man se sammenhængen mellem de forskellige relationer med deres primær- og fremmednøgler.



Figur 5.1: Relationsdiagram

## 6 Normalisering

Normaliseringen bliver bl.a. defineret ifølge Database System Concepts, [SKS11]. Her beskrives første normalform s. 327, kapitel 8, samt tredje normalform s. 336, kapitel 8. Bjarne Poulsens PowerPoint-slides, [Pou], benyttes også. De tre normalformer forklares på 4.-7. slide.

Nedenfor ses en liste over databasens tabeller med en beskrivelse af deres normalformer.

- Roller: Overholder første normalform, da hver celle består af singulære værdier, samt anden normalform, da der kun er én primærnøgle. Den overholder også tredje normalform da der ikke er nogen transitive attributter.
- Operatoer: Overholder første og anden normalform, men ikke tredje. Dette skyldes, at både opr\_id og cpr vil være unikke, og på trods af, at opr\_id er primærnøglen, vil cpr også kunne finde frem til en operatør. Tredje normalform kan opnås ved at lave en ny tabel, der eksempelvis kunne bestå af alle personlige data så som cpr, denne kunne så have opr\_id som fremmednøgle
- Raavare: Overholder første normalform, da der kun er atomiske værdier, samt anden normalform, da der kun er én primærnøgle. Den tredje normalform opfyldes også, eftersom et råvarenavn ikke kan give en leverandør og omvendt. Flere leverandører kan nemlig producere den samme råvare. Vi har altså ingen transitive attributter.
- Raavarebatch: Overholder også første normalform, da der kun er atomiske værdier, samt anden normalform, da der kun er én primærnøgle. Tredje normalform overholdes også, fordi attributten "maengde" ikke kan finde frem til et raavare\_id og omvendt. Der er altså ingen transitive attributter.
- Recept: Denne tabel overholder også alle tre normalformer. Der er kun atomiske elementer, en primærnøgle, og ingen transitive attributter.
- Receptkomponent: Overholder første normalform. Anden normalform overholdes også, da hverken recept\_id eller raavare\_id kan finde frem til nom\_netto eller tolerance alene. Tredje normalform opnås, da nom\_netto ikke kan angive tolerancen og omvendt.
- Produktbatch: Opfylder alle tre normalformer. Dette sker på samme grundlag som "Raavarebatch".
- Produktbatchkomponent: Første normalform overholdes, da der kun er singulære værdier i cellerne. Anden normalform overholdes fordi, at et produktbatch ID eller råvare ID ikke alene kan finde frem til tara,



netto, dato eller opr\_id - disse er ikke unikke, og flere operatører kan have fat i den samme produktbatchkomponent. De andre attributter kan samtidigt heller ikke finde frem til hinanden, hvilket vil sige, at tredje normalform også overholdes.

## 7 Brugere og rettigheder

### 7.1 Rollebeskrivelse

I systemet indgår fire roller; hver rolle har et ID, som gør det nemmere at overskue hvilke, roller der arver fra hvem. En rolle arver fra andre roller, som har et højere rolle ID. De fire roller står beskrevet nedenfor:

- Administrator: Denne rolle kan alt det, som de andre roller kan. Derfor har den fået rolle ID på 1. Administratoren kan oprette, ændre og læse brugere i systemet.
- Farmaceut: Denne rolle skal kunne administrere råvarer og recepter. Dette omfatter at oprette nye recepter og tilføje nye råvarer. Denne rolle har fået rolle ID 2. Rollen skal samtidigt kunne alt som Værkføreren kan.
- Værkfører: Denne rolle har mulighed for at se, hvor langt en produktbatch er nået i produktionen. Den har også mulighed for at holde styr på de forskellige råvarebatches, der er på fabrikken. Værkføreren har rolle ID 3. Denne rolle kan også alt hvad Operatøren kan.
- Operatør: Denne rolle har rolle ID'et 4, da den ikke arver fra nogen. En operatør er den rolle, som foretager selve afvejningen.

### 7.2 Rettigheder

I forhold til databasen er der én bruger. Dette er serveraccess brugeren. Denne bruger har adgang til at lave select queries på alle views, samt på tabellerne: operator og raavare. Derudover får denne bruger også rettigheder til at køre alle stored procedures (se koden under). Grunden til, at der kun er en bruger, er, at det er selve webinterfacet, der skal bruge databasen, som styrer adgangsrettighederne til forskellige dele af programmet ved hjælp af rolle tabellen.

```
Create user 'server_access'@'localhost' identified by 'qwer1234';

/* Stored procedure access rights */
Grant Execute on procedure aendre_raavare to 'server_access';
Grant Execute on procedure aendre_opr to 'server_access';
```

```
Grant Execute on procedure aendre_recept to 'server_access';
Grant Execute on procedure opret_opr to 'server_access';
Grant Execute on procedure opret_produktbatch to 'server_access';
Grant Execute on procedure opret_raavarebatch to 'server_access';
Grant Execute on procedure opret_receptkomponent to 'server_access';
Grant Execute on procedure opret_recept to 'server_access';
Grant Execute on procedure opret_raavare to 'server_access';
Grant Execute on procedure afvejning to 'server_access';

/* table and view access rights */
Grant Select on table operatoer to 'server_access';
Grant Select on table raavare to 'server_access';
Grant Select on afvejning to 'server_access';
Grant Select on antal_faerdige to 'server_access';
Grant Select on produktbatch_administration to 'server_access';
Grant Select on recept_administration to 'server_access';
Grant Select on raavarebatch_administration to 'server_access';
Grant Select on antal_komponenter to 'server_access';
```

---

## 8 Views

Vi har seks views. En oversigt samt beskrivelse af dem ses nedenfor.

1. `recept_administration`: Dette view kombinerer tabellerne: `recept`, `receptkomponent` og `raavare`. Viewet viser en tabel med recept navn, recept id og hvilke råvare der skal bruges og mængderne/tolerance af disse. Dette view skal kunne ses af alle med rolle ID 2 eller lavere, dvs. farmaceut og administrator.
2. `raavarebatch_administration`: Dette view kombinerer tabellerne: `raavarebatch` og `raavare`. Vi har for brugervenlighedens skyld valgt, at råvare navnet også skal indgå i viewet. Viewet skal vise et råvarebatch id, råvare navn og en mængde. Dette view skal kunne ses af alle med rolle ID 3 eller lavere.
3. `produktbatch_administration`: Dette view kombinerer viewsene: `antal_komponenter`, `antal_faerdige` og tabellen `recept`. For at gøre det mere overskueligt, skal dette view også vise, hvor mange færdige komponenter der er i forhold til, hvor mange komponenter recepten indeholder. Viewet indeholder ellers et recept ID, recept navn, produktbatch ID, dato for oprettelse, forholdet som beskrevet før og status på recepten. Dette view skal kunne ses med rolle ID 3 eller lavere.
4. `afvejning`: Dette view kombinerer tabellerne: `produktbatch`, `receptkomponent` og `raavare`. Viewet skal vise produktbatch ID'et, råvarebatch

ID'et, råvare navnet og mængden, der skal tilføjes. Alle, der er oprettet i systemet, kan se dette view.

5. antal\_faerdige:

---

```
create view antal_faerdige as SELECT produktbatch.pb_id,
COUNT(produktbatchkomponent.pb_id) as antal_faerdige,
recept_id, dato, status
FROM produktbatch
LEFT JOIN produktbatchkomponent on
produktbatchkomponent.pb_id=produktbatch.pb_id
group by produktbatch.pb_id;
```

---

Dette view kombinerer tabellerne: produktbatch og produktbatchkomponent. Her bliver det vist, hvor mange komponenter der er færdige i en produktbatch ved en count. Recept ID, status og dato vises også. Der er brugt et left join for, at vi også får vist produktbatches med ingen færdige komponenter.

6. antal\_komponenter: Dette view viser hvor mange komponenter der er i en recept. Det gør den ved at tælle hvor mange forskellige raavare\_id der er per recept\_id. Dette view bliver også brugt til at lave et andet view, og bliver derfor heller ikke brugt alene.

## 9 Stored Procedures

For at spare tid og reducere fejlrisici, er der oprettet en række stored procedures i databasen, som kan udføre bestemte queries.

Der er ikke mulighed for at slette noget i databasen, da hver enkelt tuple kan have store afhængigheder i resten af databasen. En råvare kan eksempelvis ikke slettes, uden at dette vil påvirke råvarebatch og receptkomponent (givet, at den pågældende råvarer altså er benyttet i disse tabeller). Et alternativ til sletning ved tastefejl, elementer, der ikke længere bliver brugt, og lignende, kunne en boolean fortælle, hvorvidt et element er aktivt. På den måde bliver de ikke slettet fra databasen, men blot sat til "deaktiveret" og vises ikke i views.

I en stored procedure, der har til opgave at ændre/opdaterer en tuple, er der naturligvis ikke mulighed for at ændre i primære nøgler.

Nedenfor ses et eksempel på en stored procedure, der opretter og ændrer recepter.

---

```
/*OPRET RECEPT */
delimiter //
create procedure opret_recept
(in id_input int(5), navn_input text)
begin
insert into recept(recept_id, recept_navn)
values (id_input, navn_input);
end; //

/*ÆNDRE RECEPT*/
create procedure aendre_recept
(in id_input int(5), navn_input text)
begin
update recept
set recept_navn = navn_input
where recept_id = id_input;
end; //
delimiter ;
```

---

Ud over Opret\_recept er der oprettet følgende stored procedures

- aendre\_recept
- opret\_opr
- aendre\_opr
- opret\_raavare
- aendre\_raavare
- opret\_raavarebatch
- opret\_produktbatch
- opret\_receptkomponent
- afvejning

De fleste procedure navne fortæller hvad de gør. Afvejning er dog lidt speciel. Afvejning opretter en produktbatchkomponent, hvorefter den tjekker om produktbatchen har en eller flere komponenter. Hvis produktbatchen har en komponent sættes status til 1. hvis alle komponenter er der, vil status blive sat til 2. Ud over dette er der også transaktionslogik i denne, se mere i [11](#).

## 10 Triggers

En trigger er en speciel form for stored procedure. Den ændrer automatisk i et view, når der for eksempel bliver indsat, slettet eller opdateret fra Java. Hvis der er flere triggers til forskellige views, kan en lille fejl i et view ændre hele databasen på meget kort tid.

Der er fordele ved at bruge triggers, da de forskellige views vil være opdateret, lige efter at der er blevet lavet en ændring.

Vi har valgt ikke at bruge triggers for at undgå, at vi laver om på hele databasen, når der eksperimenteres med forskellige views.

## 11 Transaktionslogik

Transaktionslogik bliver brugt til at sikre, at den rigtige mængde bliver taget fra råvarebatch, og tilføjet i produktbatchkomponent. Hvis mængden ikke stemmer overens, vil handlingen ikke blive gennemført. Da vi mindsker mængden i raavarebatchen og rykker det over i en produktbatchkomponent er dette nødvendigt, da en fejl i databasen vil betyde uoverensstemmelser mellem den registrerede "brugte" raavare og så den raavare der vises i raavarebatch'ene.

SQL koden for transaktionslogikken kan ses nedenfor:

---

```
create procedure afvejning
(in pb_id_input int(5), in rb_id_input int(5), in tara_input real,
in netto_input real, in opr_id_input int(5))
begin start transaction;
set @gamlemaengde = (select maengde from raavarebatch where
rb_id_input = rb_id);
set @nyemaengde = @gamlemaengde - netto_input;
update raavarebatch set maengde = @nyemaengde where
rb_id_input = rb_id;

/* OPRETTELSE AF PRODUKTbatchKOMponent */
insert into produktbatchkomponent(pb_id, rb_id, tara, netto, opr_id)
values (pb_id_input, rb_id_input, tara_input, netto_input, opr_id_input);

if(0 not in
    (select antal_komp from produktbatch_administration where
    pb_id = pb_id_input))
then update produktbatch set status=1 where pb_id=pb_id_input;
end if;

if (
```

```
(select antal_komp
  from produktbatch_administration
 where pb_id=pb_id_input) in
(select antal_faerdige
  from produktbatch_administration
 where pb_id=pb_id_input))
then update produktbatch set status=2 where pb_id=pb_id_input;
end if;

if(@gamlemaengde =
  (select maengde
    from raavarebatch
   where rb_id_input = rb_id) +
  (select netto
    from produktbatchkomponent
   where pb_id_input = pb_id and rb_id_input = rb_id)) and
(
  (select maengde
    from raavarebatch
   where rb_id_input = rb_id) >= 0)
then commit;
else rollback;
end if;
end; //
```

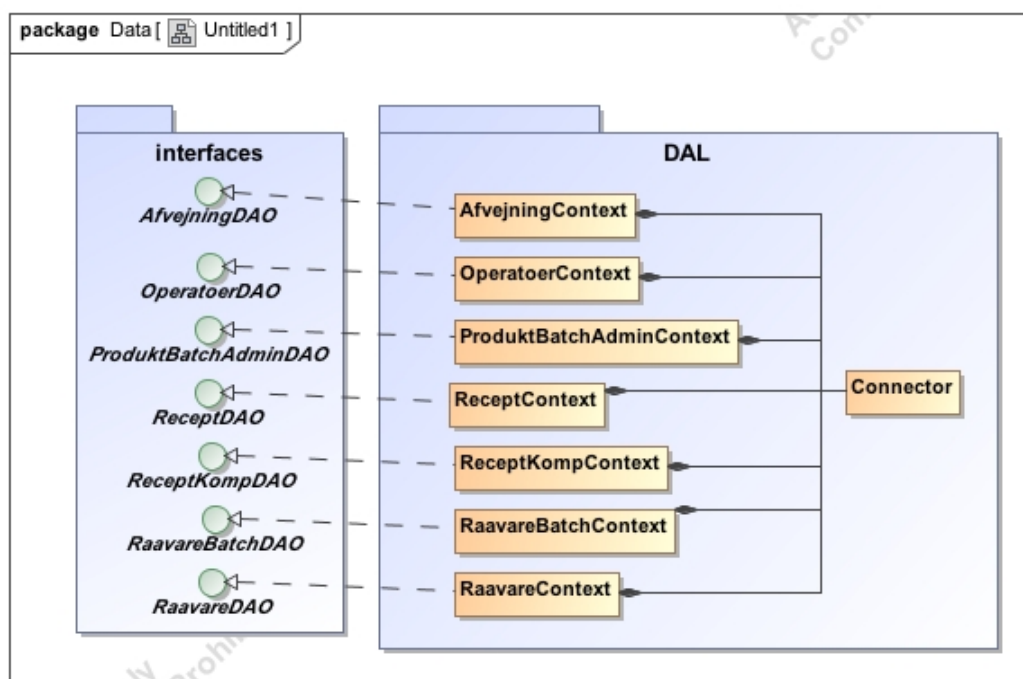
---

## 12 Applikations Interface

Applikationsinterfacet er bygget af en række interfaces kaldet xxxDAO, disse definere, hvilke funktioner der skal kunne kaldes. For eksempel har vi en operatorDAO som definerer fire funktioner.

- Hent en operatoer
- Hent alle operatoere
- Opret en operatoer
- Opdater en operatoer

Hvert DAO interface har en tilsvarende implementation, som ender på context. Disse implementerer funktionalitet i forhold til mysql databasen, og er de klasser, som kan bruges af et evt. program, til at tilgå de forskellige database operationer.



Figur 12.1: Klasse diagram

Disse context klasser bruger en connectorklasse. Denne udnytter JDBC til at kommunikere med databasen. Alt kommunikation går igennem denne, og den sørger selv for at lukke diverse kommunikationsobjekter for at undgå memory leak. Der er også oprettet en række data transfer objects. Disse bruges til at overføre data mellem et eventuelt Java program og de forskellige context klasser. Koden fylder for meget til at have i bilag, men i Appendix: Konfiguration 16.1 er der en guide til, hvordan koden tilgås på Github.

## 13 Test

Der er lavet lang række test, mange er lavet i forbindelse med at de forskellige queries er udformet. Her er tabellerne kigget igennem en efter en, for at se hvad det er for et output, vi skal forvente fra en query, herefter er en query blevet udformet, og det kan herved testes, om det er det rette output denne producere. Der er også lavet en række test af java interfacet. Dette for at sikre, at programmet virker efter hensigten. Før hver testkørsel er det vigtigt, at databasen bliver nulstillet ved at køre sql-scriptet, hvis ikke andet er nævnt i koden.

Det er alle contexts, som er blevet testet, for at se, om både stored procedures og views virker.

```
Opretter komponenter...

Create Recept...

Get Recept List:
1      margherita
2      prosciutto
3      capricciosa
7      erza

Update Recept 7:
7      olaf
```

Figur 13.1: Test af recept

På figur 13.1 ses en test af Recept. Her oprettes først to komponenter til recepten, hvorefter recepten oprettes med navnet "erza". Denne recept indeholder de to receptkomponenter, som også blev lavet. Både oprettelse af komponenter og recept fungerer, samt at hente en liste af recepter (erza er med) og opdatere ved ID. Navnet på den førhen nævnte recept ændres succesfuldt til "olaf".



## 14 Opgaver

I dette afsnit er en række queries udformet ud fra opgaverne i oplægget.

### 14.1 Query 1

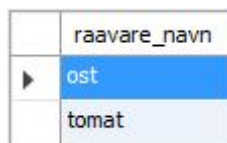
**Spørgsmål 1:** Bestem navnene på de råvarer som indgår i mindst to forskellige råvarebatches.

---

```
select raavare_navn
from raavarebatch natural join raavare
group by raavare_navn having count(raavare_navn)>1;
```

---

I denne query findes alle raavarer, som indgår i mindst to forskellige raavarebatches. Den indre forespørgsel tæller antallet af hver raavare\_id, der dukker op i raavarebatches. "Group by" grupperer optællingerne efter raavare\_id. Den ydre query vælger alle raavarer fra den indre query, hvor antallet er større end to. Resultatet ses på figur 14.1.



raavare_navn
ost
tomat

Figur 14.1: Query 1

### 14.2 Query 2

**Spørgsmål 2:** Bestem relationen som, for hver receptkomponent, indeholder tuplen (i, j, k) bestående af receptens identifikationsnummer i, receptens navn j og råvarens navn k.

---

```
select recept_id, recept_navn, raavare_navn
from recept natural join receptkomponent natural join raavare;
```

---

En meget simpel query, som bruger natural join til at sammensætte tre tabeller via deres sammenhæng af fremmednøgler. Resultatet ses på figur 14.2.

	recept_id	recept_navn	raavare_navn
►	1	margherita	dej
	1	margherita	tomat
	1	margherita	ost
	2	prosciutto	dej
	2	prosciutto	tomat
	2	prosciutto	ost
	2	prosciutto	skinke
	3	capricciosa	dej
	3	capricciosa	tomat
	3	capricciosa	ost
	3	capricciosa	skinke
	3	capricciosa	champignon

Figur 14.2: Query 2

### 14.3 Query 3

**Spørgsmål 3 del 1:** Find navnene på de recepter som indeholder mindst én af følgende to ingredienser (råvarer): skinke eller champignon.

```
select distinct recept_navn
from recept natural join receptkomponent natural join raavare
where raavare_navn = 'champignon' or raavare_navn = 'skinke';
```

En række natural joins sørger for, at vi kan få både recept\_navn samt raavare\_navn. "Distinct" bliver brugt for at sikre, at hver recept kun kommer frem én gang. Hvis recepten både har "skinke" og "champignon" ville der ellers komme to rækker med samme receptnavn. "Where" klausulen sørger for, at vi kun får de recepter, som har receptkomponenter, der indeholder enten skinke eller champignon. Resultatet ses på figur 14.3.

	recept_navn
►	prosciutto
	capricciosa

Figur 14.3: Query 3 del 1

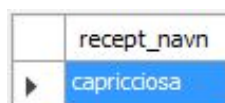
**Spørgsmål 3 del 2:** Find navnene på de recepter som indeholder både ingrediensen skinke og ingrediensen champignon.

---

```
select distinct recept_navn
from recept natural join receptkomponent natural join raavare
where raavare_navn = 'skinke' and recept_navn in
      (select recept_navn
       from recept natural join receptkomponent natural join raavare
       where raavare_navn = 'champignon');
```

---

Denne query finder en recept som indeholder både skinke og champignon, ved at lave en indre query som finder recepter som indeholder champignon herefter, hvis recepten er i denne tabel og også indeholder ingrediensen skinke, bliver den vist i tabellen. Resultatet ses på figur 14.4.



recept_navn
capricciosa

Figur 14.4: Query 3 del 2

#### 14.4 Query 4

**Spørgsmål 4:** Find navnene på de recepter som *ikke* indeholder ingrediensen champignon.

---

```
select distinct recept_navn
from recept
where recept_id not in(
      select recept_id
      from receptkomponent natural join raavare
      where raavare_navn = 'champignon');
```

---

Den indre query finder alle recept\_id'er som indeholder champignon, mens den ydre vælger alle recepter, hvor recept\_id ikke er i den indre query. På denne måde bliver alle med champignon sorteret fra. Resultatet ses på figur 14.5.

	recept_navn
►	margherita
	prosciutto

Figur 14.5: Query 4

## 14.5 Query 5

**Spørgsmål 5:** Find navnene på den eller de recepter som indeholder den største nominelle vægt af ingrediensen tomat.

```
select recept_navn, sum(nom_netto) as amount
from recept natural join receptkomponent
      natural join raavare
where raavare_navn = 'tomat'
group by recept_id
having sum(nom_netto) = (select Max(amount)
                        from (select recept_id, sum(nom_netto) as amount
                              from receptkomponent natural join raavare
                              where raavare_navn = 'tomat'
                              group by recept_id) t);
```

Antagelsen i denne query er, at det er den samlede vægt af `nom_netto` af alle typer tomat i en recept. Dette er valgt grundet formuleringen af opgaven. Da databasen er designet på en sådan måde, at forskellige varianter af tomat godt kan være i en enkelt opskrift, bliver vi nødt til at summere den nominelle vægt af alle tomat ingredienser i en recept, før vi finder den/de største.

De første fem linjer af denne query vælger alle receptnavne og deres sum af `nom_netto` af tomat i deres receptkomponenter. Fra linje 6 bliver der brugt en "having" klausul, som fortæller, at vi kun vil have de recepter, hvor summen af vægten er lig max vægt. Resultatet ses på figur 14.6.

	recept_navn	amount
►	margherita	2
	prosciutto	2

Figur 14.6: Query 5

## 14.6 Query 6

**Spørgsmål 6:** Bestem relationen som for hver produktbatchkomponent indeholder tuplen (i, j, k) bestående af produktbatchets identifikationsnummer i, råvarens navn j og råvarens nettovægt k.

---

```
select pb_id, raavare_navn, netto
from produktbatchkomponent natural join raavarebatch natural join raavare;
```

---

En simple query, som bruger natural joins til at sammensætte en række tabeller ved hjælp af deres fremmednøgler. Resultatet ses på figur 14.7.

	pb_id	raavare_navn	netto
▶	1	dej	10.05
	1	tomat	2.03
	1	ost	1.98
	2	dej	10.01
	2	tomat	1.99
	2	ost	1.47
	3	dej	10.07
	3	tomat	2.06
	3	ost	1.55
	3	skinke	1.53
	4	dej	10.02
	4	ost	1.57
	4	skinke	1.03
	4	champignon	0.99

Figur 14.7: Query 6

## 14.7 Query 7

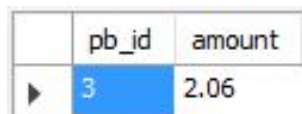
**Spørgsmål 7:** Find identifikationsnumrene af den eller de produktbatches som indeholder en størst nettovægt af ingrediensen tomat.

---

```
select pb_id, sum(netto) as amount
from produktbatchkomponent natural join raavarebatch natural join raavare
where raavare_navn = 'tomat'
group by pb_id
having sum(netto) = (select Max(amount)
                    from (select pb_id, sum(netto) as amount
                          from produktbatchkomponent
                          natural join raavarebatch
                          natural join raavare
                          where raavare_navn = 'tomat'
                          group by pb_id) t);
```

---

Query 5 fungerer på samme måde. Her bruges produktbatch og produktbatchkomponent i stedet. Resultatet ses på figur 14.8.



	pb_id	amount
▶	3	2.06

Figur 14.8: Query 7

## 14.8 Query 8

**Spørgsmål 8:** Find navnene på alle operatører som har været involveret i at producere partier af varen margherita.

---

```
select distinct opr_id, opr_navn
from operatoer natural join produktbatchkomponent
              natural join recept
              natural join produktbatch
where recept_navn = 'margherita';
```

---

Denne query bruger tre natural joins til at sætte fire tabeller sammen ved hjælp af fremmednøgler. Ud fra dette, bliver der valgt operatør navnet og operatør id, hvor pizzaens navn er lig margherita. Resultatet ses på figur 14.9.

	opr_id	opr_navn
▶	1	Angelo A
	2	Antonella B

Figur 14.9: Query 8

## 14.9 Query 9

**Spørgsmål 9:** Bestem relationen som for hver produktbatchkomponent indeholder tuplen (i, j, k, l, m, n) bestående af produktbatchets identifikationsnummer i, produktbatchets status j, råvarens navn k, råvarens nettovægt l, den tilhørende receipts navn m og operatørens navn n.

```
select pb_id, status, raavare_navn, netto, receipt_navn, opr_navn
from produktbatchkomponent
    natural join produktbatch
    natural join raavarebatch
    natural join raavare
    natural join receipt
    natural join operatoer;
```

En simpel query som bruger natural join til at sammensætte en række tabeller ved hjælp af Resultatet ses på figur 14.10.

	pb_id	status	raavare_navn	netto	receipt_navn	opr_navn
▶	1	2	dej	10.05	margherita	Angelo A
	1	2	tomat	2.03	margherita	Angelo A
	1	2	ost	1.98	margherita	Angelo A
	2	2	dej	10.01	margherita	Antonella B
	2	2	tomat	1.99	margherita	Antonella B
	2	2	ost	1.47	margherita	Antonella B
	3	2	dej	10.07	prosciutto	Angelo A
	3	2	tomat	2.06	prosciutto	Antonella B
	3	2	ost	1.55	prosciutto	Angelo A
	3	2	skinke	1.53	prosciutto	Antonella B
	4	1	dej	10.02	capricciosa	Luigi C
	4	1	ost	1.57	capricciosa	Luigi C
	4	1	skinke	1.03	capricciosa	Luigi C
	4	1	champignon	0.99	capricciosa	Luigi C

Figur 14.10: Query 9

## 15 Konklusion

Databasen der er blevet udleveret, er blevet analyseret, og der er blevet tilføjet en ny tabel som det blev vurderet der manglede. Databasen er blev tilføjet views, stored procedures og transactionlogik, og der er oprettet en bruger som har begrænset adgang. Herved er der opnået en sikkerhed, ved at brugere ikke kan indsætte ødelæggende data, eller fjerne tabeller. Al oprettelse af tupler foregår igennem stored procedures, og størstedelen af select statements bliver lavet gennem views. Enkelte tabeller er blevet vurderet til at kunne læses direkte. Der er oprettet et java applikationsinterface, som kan udnyttets af andre java programmer til at tilgå databasen.



## 16 Appendiks: Konfiguration

### 16.1 Klon Repository

Det færdige program er lagt op på github og kan hentes derfra

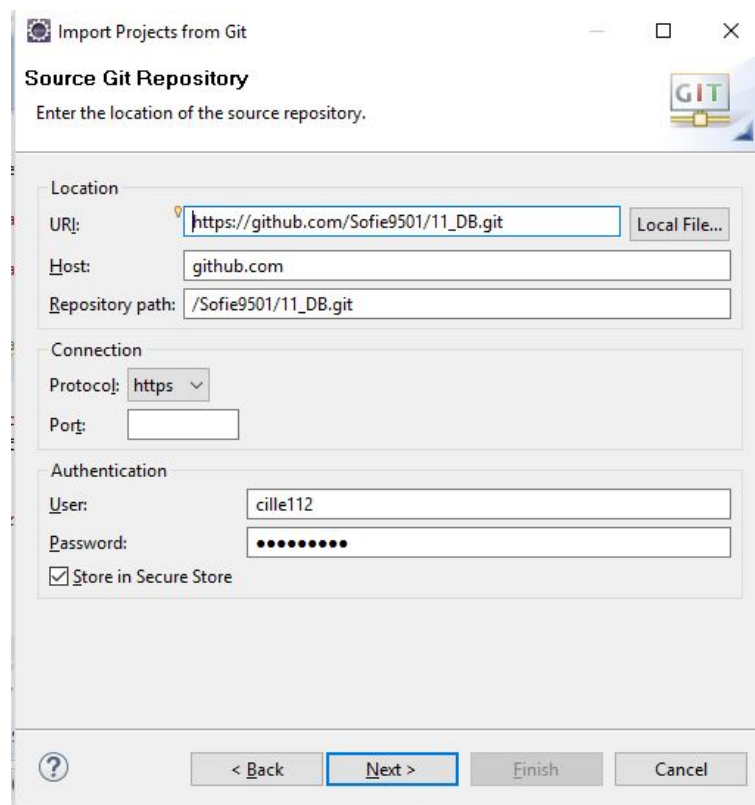
Koden kan ses direkte i browser på linket [https://github.com/Sofie9501/11\\_DB](https://github.com/Sofie9501/11_DB)

eller det kan hentes ved følgende: I Eclipse skal der vælges at importere og klon fra git.

[https://github.com/Sofie9501/11\\_DB.git](https://github.com/Sofie9501/11_DB.git) skal indsættes på URI feltets plads.

I Host feltet indtastes "Github.com" og i Repository path indtastes "/Sofie9501/11\_DB.git".

Når alt er indtastet, skulle vinduet gerne se ud som på figur 16.1 dog med en anden user og password.



Figur 16.1: Screenshot af hvordan importvinduet skal se ud

### 16.2 PC konfiguration

Programmet er kørt og testet på følgende opsætning

- Styresystem: Windows 10 64 bit.
- CPU: Intel Core I5-4300U 1.9 Ghz

- RAM: 4 GB
- Java jdk version 1.8.0-u60
- Java version 8 Update 60
- Eclipse Mars Version: Mars Release (4.5.0), Build id: 20150621-1200
- MYSQL version: 5.7.10

## 17 Bilag

### 17.1 SQL kode

```
/* must be dropped in this order to avoid constraint violations */
DROP TABLE IF EXISTS produktbatchkomponent;
DROP TABLE IF EXISTS produktbatch;
DROP TABLE IF EXISTS operatoer;
DROP TABLE IF EXISTS receptkomponent;
DROP TABLE IF EXISTS recept;
DROP TABLE IF EXISTS raavarebatch;
DROP TABLE IF EXISTS raavare;
DROP TABLE IF EXISTS roller;

CREATE TABLE roller(rolle_id INT PRIMARY KEY, rolle_navn TEXT) ENGINE = innnoDB;

CREATE TABLE operatoer(opr_id INT PRIMARY KEY, opr_navn TEXT, cpr TEXT,
password TEXT, rolle_id INT,
FOREIGN KEY (rolle_id) REFERENCES roller(rolle_id)) ENGINE=innnoDB;

CREATE TABLE raavare(raavare_id INT PRIMARY KEY, raavare_navn TEXT,
leverandoer TEXT) ENGINE=innnoDB;

CREATE TABLE raavarebatch(rb_id INT PRIMARY KEY, raavare_id INT, maengde REAL,
FOREIGN KEY (raavare_id) REFERENCES raavare(raavare_id)) ENGINE=innnoDB;

CREATE TABLE recept(recept_id INT PRIMARY KEY, recept_navn TEXT) ENGINE=innnoDB;

CREATE TABLE receptkomponent(recept_id INT, raavare_id INT, nom_netto REAL,
tolerance REAL,
PRIMARY KEY (recept_id, raavare_id),
FOREIGN KEY (recept_id) REFERENCES recept(recept_id),
FOREIGN KEY (raavare_id) REFERENCES raavare(raavare_id)) ENGINE=innnoDB;

CREATE TABLE produktbatch(pb_id INT PRIMARY KEY, status INT, recept_id INT,
dato DATE,
FOREIGN KEY (recept_id) REFERENCES recept(recept_id)) ENGINE=innnoDB;

CREATE TABLE produktbatchkomponent(pb_id INT, rb_id INT, tara REAL,
netto REAL, opr_id INT,
PRIMARY KEY (pb_id, rb_id),
FOREIGN KEY (pb_id) REFERENCES produktbatch(pb_id),
FOREIGN KEY (rb_id) REFERENCES raavarebatch(rb_id),
```

```
FOREIGN KEY (opr_id) REFERENCES operatoer(opr_id)) ENGINE=innnoDB;

INSERT INTO roller(rolle_id, rolle_navn) VALUES
(1, 'DBA'),
(2, 'Farmaceut'),
(3, 'Værkfører');

INSERT INTO operatoer(opr_id, opr_navn, rolle_id, cpr, password) VALUES
(1, 'Angelo A', 1, '070770-7007', 'lKje4fa'),
(2, 'Antonella B', 2, '080880-8008', 'atoJ21v'),
(3, 'Luigi C', 3, '090990-9009', 'jEfm5aQ');

INSERT INTO raavare(raavare_id, raavare_navn, leverandoer) VALUES
(1, 'dej', 'Wawelka'),
(2, 'tomat', 'Knoor'),
(3, 'tomat', 'Veaubais'),
(4, 'tomat', 'Franz'),
(5, 'ost', 'Ost og Skinke A/S'),
(6, 'skinke', 'Ost og Skinke A/S'),
(7, 'champignon', 'Igloo Frostvarer');

INSERT INTO raavarebatch(rb_id, raavare_id, maengde) VALUES
(1, 1, 1000),
(2, 2, 300),
(3, 3, 300),
(4, 5, 100),
(5, 5, 100),
(6, 6, 100),
(7, 7, 100);

INSERT INTO recept(recept_id, recept_navn) VALUES
(1, 'margherita'),
(2, 'prosciutto'),
(3, 'capricciosa');

INSERT INTO receptkomponent(recept_id, raavare_id, nom_netto, tolerance) VALUES
(1, 1, 10.0, 0.1),
(1, 2, 2.0, 0.1),
(1, 5, 2.0, 0.1),

(2, 1, 10.0, 0.1),
(2, 3, 2.0, 0.1),
(2, 5, 1.5, 0.1),
```

```
(2, 6, 1.5, 0.1),
```

```
(3, 1, 10.0, 0.1),  
(3, 4, 1.5, 0.1),  
(3, 5, 1.5, 0.1),  
(3, 6, 1.0, 0.1),  
(3, 7, 1.0, 0.1);
```

```
INSERT INTO produktbatch(dato,pb_id, recept_id, status) VALUES  
(NOW(),1, 1, 2),  
(NOW(),2, 1, 2),  
(NOW(),3, 2, 2),  
(NOW(),4, 3, 1),  
(NOW(),5, 3, 0);
```

```
INSERT INTO produktbatchkomponent(pb_id, rb_id, tara, netto, opr_id) VALUES  
(1, 1, 0.5, 10.05, 1),  
(1, 2, 0.5, 2.03, 1),  
(1, 4, 0.5, 1.98, 1),  
  
(2, 1, 0.5, 10.01, 2),  
(2, 2, 0.5, 1.99, 2),  
(2, 5, 0.5, 1.47, 2),  
  
(3, 1, 0.5, 10.07, 1),  
(3, 3, 0.5, 2.06, 2),  
(3, 4, 0.5, 1.55, 1),  
(3, 6, 0.5, 1.53, 2),  
  
(4, 1, 0.5, 10.02, 3),  
(4, 5, 0.5, 1.57, 3),  
(4, 6, 0.5, 1.03, 3),  
(4, 7, 0.5, 0.99, 3);
```

```
use lab;  
/*****  
* VIEWS  
*****/  
Drop view if exists recept_administration;  
Drop view if exists raavarebatch_administration;  
Drop view if exists antal_faerdige;  
Drop view if exists antal_komponenter;  
Drop view if exists produktbatch_administration;
```

\*

```
Drop view if exists afvejning;
```

```
create view recept_administration
as select recept_navn, recept_id, raavare_navn, raavare_id, tolerance, nom_netto
from recept natural join receptkomponent natural join raavare;
```

```
create view raavarebatch_administration
as select rb_id, raavare_navn, maengde, raavare_id
from raavare natural join raavarebatch;
```

```
create view antal_faerdige as SELECT produktbatch.pb_id,
COUNT(produktbatchkomponent.pb_id) as antal_faerdige, recept_id, dato, status
FROM produktbatch
LEFT JOIN produktbatchkomponent on
produktbatchkomponent.pb_id=produktbatch.pb_id group by produktbatch.pb_id;
```

```
create view antal_komponenter as
select recept_id, count(raavare_id) as antal_komp
from receptkomponent group by recept_id;
```

```
create view produktbatch_administration as select pb_id, recept_id,
recept_navn, antal_komp, antal_faerdige, dato, status
from antal_komponenter natural join antal_faerdige natural join recept;
```

```
create view afvejning as select pb_id, raavare_id , raavare_navn,
nom_netto, tolerance
from produktbatch natural join receptkomponent natural join raavare
where (receptkomponent.raavare_id, produktbatch.pb_id) not in(
select raavare_id, pb_id from produktbatchkomponent natural join raavarebatch);
```

```
/******
```

```
* STORED PROCEDURES
```

```
*
```

```
*****/*
```

```
drop procedure if exists opret_opr;
drop procedure if exists opret_raavare;
drop procedure if exists aendre_raavare;
DROP PROCEDURE IF EXISTS opret_recept;
DROP PROCEDURE IF EXISTS aendre_recept;
DROP PROCEDURE IF EXISTS opret_raavarebatch;
```

```
DROP PROCEDURE IF EXISTS opret_produktbatch;
DROP PROCEDURE IF EXISTS opret_receptkomponent;
DROP PROCEDURE IF EXISTS aendre_opr;
drop procedure if exists afvejning;

delimiter //

/* OPRET OPERATOR */
create procedure opret_opr
(in id_input int(5), in rolle_input int(1),
in navn_input text, in cpr_input text, in password_input text)
begin
insert into operatoer(opr_id, rolle_id, opr_navn, cpr, password)
values (id_input, rolle_input, navn_input, cpr_input, password_input);
end; //

/* AENDRE RAAVARE */
create procedure aendre_opr
(in id_input int(5), in rolle_input int(1), in navn_input text,
in cpr_input text, in password_input text)
begin
update operatoer
set opr_navn = navn_input, rolle_id = rolle_input, cpr = cpr_input,
password = password_input
where opr_id = id_input;
end; //

/* OPRET RAAVARE */
create procedure opret_raavare
(in raavare_id_input int, in raavare_navn_input text, in leverandoer_input text)
begin
insert into raavare(raavare_id, raavare_navn, leverandoer)
values (raavare_id_input, raavare_navn_input, leverandoer_input);
end; //

/* AENDRE RAAVARE */
create procedure aendre_raavare
(in raavare_id_input int, in raavare_navn_input text, in leverandoer_input text)
begin
update raavare
set raavare_navn = raavare_navn_input, leverandoer = leverandoer_input
where raavare_id = raavare_id_input;
end; //
```

```
/* MANGLER SLET RAAVARE */

/*OPRET RECEPT */
create procedure opret_recept
(in id_input int, in navn_input text)
begin
insert into recept(recept_id, recept_navn)
values (id_input, navn_input);
end; //

/*ENDRE RECEPT*/
create procedure aendre_recept
(in id_input int(5), in navn_input text)
begin
update recept
set recept_navn = navn_input
where recept_id = id_input;
end; //

/*OPRET RAAVAREBATCH */
create procedure opret_raavarebatch
(in rb_id_input int(5), in rv_id_input int(5), in maengde_input real)
begin
insert into raavarebatch(rb_id, raavare_id, maengde)
values (rb_id_input, rv_id_input, maengde_input);
end; //

/*OPRET PRODUKTBATCH */
create procedure opret_produktbatch
(in pb_id_input int(5), in status_input int(2), in recept_id_input int(5))
begin
insert into produktbatch(pb_id, status, recept_id, dato)
values (pb_id_input, status_input, recept_id_input, NOW());
end; //

/*OPRET RECEPTKOMPLEMENT*/
create procedure opret_receptkomponent
(in recept_id_input int(5), in raavare_id_input int(5), in netto_input real,
in tolerance_input real)
begin
```



```
insert into receptkomponent(recept_id, raavare_id, nom_netto, tolerance)
values (recept_id_input, raavare_id_input, netto_input, tolerance_input);
end; //
```

```
/*AFVEJNING*/
create procedure afvejning
(in pb_id_input int(5), in rb_id_input int(5), in tara_input real,
in netto_input real, in opr_id_input int(5))
begin start transaction;
set @gamlemaengde = (select maengde from raavarebatch
where rb_id_input = rb_id);
set @nyemaengde = @gamlemaengde - netto_input;
update raavarebatch set maengde = @nyemaengde
where rb_id_input = rb_id;
```

```
/* OPRETTELSE AF PRODUKTbatchKOMponent */
insert into produktbatchkomponent(pb_id, rb_id, tara, netto, opr_id)
values (pb_id_input, rb_id_input, tara_input, netto_input, opr_id_input);
```

```
if(0 not in (select antal_komp from produktbatch_administration
where pb_id=pb_id_input))
then update produktbatch set status=1
where pb_id=pb_id_input;
end if;
```

```
if ((select antal_komp from produktbatch_administration where pb_id=pb_id_input)
in (select antal_faerdige from produktbatch_administration
where pb_id=pb_id_input))
then update produktbatch set status=2 where pb_id=pb_id_input;
end if;
```

```
if(@gamlemaengde = (select maengde from raavarebatch where rb_id_input = rb_id) +
(select netto from produktbatchkomponent where pb_id_input = pb_id
and rb_id_input = rb_id)) and
((select maengde from raavarebatch where rb_id_input = rb_id) >= 0)
then commit;
else rollback;
end if;
end; //
```

```
delimiter ;
```

```
/* User rights */
drop user 'server_access'@'localhost';
create user 'server_access'@'localhost' identified by 'qwer1234';

/* Stored procedure access rights */
Grant Execute on procedure aendre_raavare to 'server_access';
Grant Execute on procedure aendre_opr to 'server_access';
Grant Execute on procedure aendre_recept to 'server_access';
Grant Execute on procedure opret_opr to 'server_access';
Grant Execute on procedure opret_produktbatch to 'server_access';
Grant Execute on procedure opret_raavarebatch to 'server_access';
Grant Execute on procedure opret_receptkomponent to 'server_access';
Grant Execute on procedure opret_recept to 'server_access';
Grant Execute on procedure opret_raavare to 'server_access';
Grant Execute on procedure afvejning to 'server_access';

/* table and view access rights */
Grant Select on table operatoer to 'server_access';
Grant Select on table raavare to 'server_access';
Grant Select on afvejning to 'server_access';
Grant Select on antal_faerdige to 'server_access';
Grant Select on produktbatch_administration to 'server_access';
Grant Select on recept_administration to 'server_access';
Grant Select on raavarebatch_administration to 'server_access';
Grant Select on antal_komponenter to 'server_access';
```

---

## 18 Litteraturliste

- [CDI16] CDIO. *CDIO Final*. [https://docs.google.com/document/d/1Bcq1GHdgZh79TnQQqW2n3T3p8rwQBewad-J3p\\_6HWoE/edit](https://docs.google.com/document/d/1Bcq1GHdgZh79TnQQqW2n3T3p8rwQBewad-J3p_6HWoE/edit). [Online; hentet 29-Marts-2016]. 2016.
- [Pou] Bjarne Poulsen. *Normalization*. <https://drive.google.com/file/d/0B8wd6ieXWsQOY21UenBYazQ2eGc/view?usp=sharing>. [Online; hentet 29-marts-2016].
- [SKS11] Abraham Silberschatz, Henry F. Korth og S. Sudarshan. *Database System Concepts*. 6. udg. .: McGraww Hill Inc., 1221 Avenue of the Americas, New York, NY 10020, ISBN: 978-0-07-352332-3, 2011.