**Auteurs:** Thomas Daane, Marcel Forman, Bart Frijters & Wesley Geboers

**Datum:** 10-02-2022

**Plaats:** Breda

**Jaar:** 2022

SilentDisco

Beheersdocument

Contents

[1. Algemeen 1](#_Toc99039106)

[2. Aanpak 1](#_Toc99039107)

[2.1 Planning 1](#_Toc99039108)

[2.2 Afspraken 2](#_Toc99039109)

[3. Functional requirements 2](#_Toc99039110)

[3.1 Algemeen 2](#_Toc99039111)

[3.2 Webshop 3](#_Toc99039112)

[3.3 Klantenportaal 3](#_Toc99039113)

[3.4 Medewerkersportaal 4](#_Toc99039114)

[4. Entity Relations Diagram (ERD) 6](#_Toc99039115)

[4.1 Toelichting ERD 6](#_Toc99039116)

[4.1.1 Employees 6](#_Toc99039117)

[4.1.2 Roles 7](#_Toc99039118)

[4.1.3 Products 7](#_Toc99039119)

[4.1.4 ProductLogs 7](#_Toc99039120)

[4.1.5 Users 7](#_Toc99039121)

[5. Relational model (RD) 8](#_Toc99039122)

[5.1 Toelichting RD 8](#_Toc99039123)

[5.1.1 Employees 8](#_Toc99039124)

[5.1.2 Roles 8](#_Toc99039125)

[5.1.3 Employee-Roles 8](#_Toc99039126)

[5.1.4 Products 9](#_Toc99039127)

[5.1.5 ProductLogs 9](#_Toc99039128)

[5.1.6 Users 9](#_Toc99039129)

[5.1.7 OrderHeaders 9](#_Toc99039130)

[5.1.8 OrderLines 9](#_Toc99039131)

[6. Database realisatie (DDL) 10](#_Toc99039132)

[6.1 Tables 10](#_Toc99039133)

[6.1.1 Structuur table Employees 10](#_Toc99039134)

[6.1.2 Structuur table Roles 11](#_Toc99039135)

[6.1.3 Structuur table Employees-Roles 12](#_Toc99039136)

[6.1.4 Structuur table Products 13](#_Toc99039137)

[6.1.5 Structuur table ProductLogs 14](#_Toc99039138)

[6.1.6 Structuur table Users 15](#_Toc99039139)

[6.1.7 Structuur table OrderHeaders 16](#_Toc99039140)

[6.1.8 Structuur table OrderLines 17](#_Toc99039141)

[6.2 Triggers 18](#_Toc99039142)

[6.2.1 Trigger LineTotalPrice 18](#_Toc99039143)

[6.2.2 Trigger UpdateTotalPriceHeader 18](#_Toc99039144)

[6.3 Views 19](#_Toc99039145)

[6.3.1 View Top10ProductsAmount 19](#_Toc99039146)

[6.3.2 View Top10ProductsPrice 20](#_Toc99039147)

[6.3.3 View RevenuePerMonth 20](#_Toc99039148)

[7. Security (DCL) 21](#_Toc99039149)

[7.1 Admin 21](#_Toc99039150)

[7.2 system 21](#_Toc99039151)

[8. Maintenance 22](#_Toc99039152)

[8.1 Instructie backup 22](#_Toc99039153)

[8.2 Instructie analyze table 22](#_Toc99039154)

[8.3 Instructie Check table 22](#_Toc99039155)

[8.3 Instructie optimize table 22](#_Toc99039156)

[9. Installatie 23](#_Toc99039157)

[9.2 Installatie handleiding 23](#_Toc99039158)

[9.3 Insert data 23](#_Toc99039159)

[9.3.1 Insert Employees 23](#_Toc99039160)

[9.3.2 Insert Users 23](#_Toc99039161)

[9.3.3 Insert Roles 23](#_Toc99039162)

[9.3.4 Insert Employees-Roles 24](#_Toc99039163)

[9.3.5 Insert Products 24](#_Toc99039164)

[9.3.6 Insert ProductLogs 24](#_Toc99039165)

[9.3.7 Insert OrderHeaders 25](#_Toc99039166)

[9.3.8 Insert OrderLines 25](#_Toc99039167)

[10 Uitwerking (DML) 26](#_Toc99039168)

[10.1 Edit user 26](#_Toc99039169)

[10.2 Show orders 26](#_Toc99039170)

[10.3 Show order details 27](#_Toc99039171)

[10.4 Edit products 27](#_Toc99039172)

# Algemeen

In dit document zal de database voor StilteAUBv gedocumenteerd worden. StilteAUBv heeft momenteel een promotionele website die omgebouwd zal worden naar een webshop. Om de webshop goed te laten functioneren zal er een database gebouwd en gekoppeld moeten worden.

Vanuit de webshop is het voor de klant mogelijk om bestellingen te doen en inzien. Dit betekent dat de database hier op voorbereid moet zijn. De database zal daardoor data moeten kunnen vastleggen van bijvoorbeeld artikelen, bestellingen en klanten.

Ook is zal de webshop voorzien worden van een medewerkersportaal waaruit op basis van verschillende rechten artikelen aangemaakt en gewijzigd kunnen worden. Ook zal het vanuit hier mogelijk zijn om de gemaakte bestellingen in te zien en van status te veranderen. Omdat niet alle data beschikbaar mag zijn voor alle medewerkers zullen er rollen en rechten gemaakt moeten worden. De database zal om deze reden ook data moeten kunnen vastleggen voor medewerkers en rollen.

Met dit in gedachte zal het vervolg van dit document de database structuur verder uitwerken. Zo zullen er ERD en RD tekeningen gemaakt worden, zullen de query’s gedocumenteerd worden en zal de security en onderhoud verder beschreven worden.

# Aanpak

## 2.1 Planning

Om dit project in goede banen te kunnen leiden is er een globale planning gemaakt. De verschillende stappen zijn beschreven met de daar bijhorende verantwoordelijke en start en eind datum. De planning is hieronder te zien.

|  |  |  |  |
| --- | --- | --- | --- |
| Beschrijving | Verantwoordelijke | Start | Eind |
| Start Periode 3 |  | 02-02-2022 |  |
| Beheersdocument – 1 Algemeen | Thomas Daane | 09-02-2022 | 17-02-2022 |
| Beheersdocument – 2 Aanpak | Wesley Geboers | 09-02-2022 | 17-02-2022 |
| Beheersdocument – 3 Functional requirements | Team | 09-02-2022 | 17-02-2022 |
| Beheersdocument – 4 ERD | Team | 09-02-2022 | 17-02-2022 |
| Beheersdocument – 5 Relational model | Team | 09-02-2022 | 17-02-2022 |
| Beheersdocument – 6 Database realization | Team | 18-02-2022 | 10-03-2022 |
| Beheersdocument – 7 Security | Marcel Forman | 18-02-2022 | 10-03-2022 |
| Beheersdocument – 8 Maintanance | Bart Frijters | 18-02-2022 | 10-03-2022 |
| Github inrichten | Wesley Geboers | 17-02-2022 | 18-02-2022 |
| Database inrichten | Team | 18-02-2022 | 10-03-2022 |
| webbased frontend inrichten | Team | 10-03-2022 | 17-03-2022 |
| Oplevering |  |  | 27-03-2022 |

## 2.2 Afspraken

Voor het schrijven van dit document, versiebeheer en query’s zijn er als team een aantal afspraken gemaakt. Dit heeft als doel om uniform te werk te gaan. Hieronder zijn de verschillende afspraken beschreven.

|  |  |
| --- | --- |
| Definitie | Toelichting |
| Versiebeheer | Versiebeheer zal plaatsvinden op GitHub. |
| Taal | De tabellen, kolommen etc. worden geschreven in het engels. |
| Query’s | * Sleutelwoorden in hoofdletters. Denk hierbij aan SELECT en WHERE. * Tabelnamen in meervoud. * Kolomnamen in enkelvoud. * Tekst, datum en tijd in enkele aanhalingstekens. * Getallen zonder aanhalingstekens. |
| Modellen | De modellen worden getekend op Draw.io |

# Functional requirements

## 3.1 Algemeen

**Scenario 1: Registreren**

**User story:** De gebruiker heeft de mogelijkheid om een account aan te maken voor het inzien van zijn of haar bestellingen en de daarbij horende statussen.

**Scenario:** De gebruiker bezoekt de site, heeft nog geen account en beslist een account aan te maken.

**Given:** De gebruiker heeft nog geen account en drukt op de knop registreren.

**When:** De gebruiker heeft op de knop registreren gedrukt waarna er een formulier verschijnt.

**And:** De gebruiker vult de benodigde gegevens in.

**Then:** De gebruiker drukt op opslaan waarna de gegevens worden opgeslagen in de tabel `Users`.

**Scenario 2: Inloggen**

**User story:** De gebruiker kan indien hij of zij een account heeft inloggen op de website om zo zijn of haar bestellingen en de daarbij horende statussen in te zien.

**Scenario:** De gebruiker bezoekt de site en wilt de status van zijn of haar bestelling weten.

**Given:** De gebruiker heeft een bestaand account en drukt op de knop inloggen.

**When:** De gebruiker heeft op de knop inloggen gedrukt en vult zijn of haar inloggegevens in.

**And:** De site haalt toont de profiel pagina van de gebruiker. Deze gegevens worden aan de hand van het account opgehaald uit de database en getoond op de pagina.

**Then:** De gebruiker drukt op de knop bestellingen waarna de bestellingen van de ingelogde gebruiker worden opgehaald uit de database en getoond op de pagina bestellingen.

## 3.2 Webshop

**Scenario 1: Producten toevoegen aan winkelmandje**

**User story:** De gebruiker kan op de website 1 of meerdere producten aan zijn of haar winkelmandje toevoegen om later te bestellen.

**Scenario:** De gebruiker bezoekt de site en ziet een product die hij of zij wilt bestellen.

**Given:** De gebruiker ziet een product die hij of zij wilt bestellen, vult een aantal in en drukt op de knop toevoegen.

**When:** De gebruiker heeft een aantal ingevuld en op de knop toevoegen gedrukt.

**Then:** De producten zullen tijdelijk worden opgeslagen in de cache en toegevoegd aan het winkelmandje.

**Scenario 2: Bestellen**

**User story:** De gebruiker heeft 1 of meerdere producten toegevoegd aan het winkelmandje en kan deze producten nu bestellen.

**Scenario:** De gebruiker heeft 1 of meerdere producten aan het winkelmandje toegevoegd en wilt deze gaan bestellen.

**Given:** De gebruiker heeft alle gewenste producten en aantallen ingevuld en drukt op de knop bestellen.

**When:** De gebruiker heeft op de knop bestellen gedrukt en krijgt een popup te zien met alle producten die besteld worden en het totaal bedrag. De gebruiker kiest een betaal methode en drukt op de knop bestellen.

**And:** De gebruiker betaald via de ideal pagina.

**Then:** De gebruiker word doorverwezen naar een pagina waarop word aangegeven dat de bestelling is gelukt. Gelijker tijd word de bestelling opgeslagen in de tabellen `OrderHeaders` en `OrderLines`.

## 3.3 Klantenportaal

**Scenario 1: Profielgegevens wijzigen**

**User story:** De gebruiker heeft de mogelijkheid om zijn of haar profielgegevens te wijzigen in het klanten portaal.

**Scenario:** De gebruiker heeft een bestaand account en wilt zijn of haar gegevens wijzigen.

**Given:** De gebruiker heeft al een account, is ingelogd en drukt op de knop profiel.

**When:** De gebruiker heeft op de knop profiel gedrukt waarna de data van de ingelogde gebruiker word opgehaald vanuit de database en getoond kan worden op de profiel pagina.

**And:** De gebruiker drukt op de knop profielgegevens wijzigen, past de gewenste gegevens aan en drukt op opslaan.

**Then:** De gewijzigde gegevens zullen worden opgeslagen in de table `Users`. Er zal op dat moment een update plaatsvinden op de record die bij het ingelogde account hoort.

## 3.4 Medewerkersportaal

**Scenario 1: Medewerker aanmaken**

**User story:** IT medewerkers hebben de rechten om accounts voor medewerkers van StilteAUBv aan te maken.

**Scenario:** De IT medewerkers heeft een bestaand account, de benodigde rechten en wilt een account aanmaken voor een medewerker van StilteAUBv.

**Given:** De IT medewerker heeft een bestaand account, is onderdeel van de rol IT, zit in het medewerkersportaal en drukt op de knop account aanmaken.

**When:** De IT medewerker heeft op de knop account aanmaken gedrukt waarna de site een formulier opent voor het aanmaken van de account.

**And:** De gebruiker vult de benodigde gegevens in en drukt op de knop opslaan.

**Then:** De gegevens zullen worden opgeslagen in de tabel `Employees`.

**Scenario 2: Rechten wijzigen**

**User story:** IT medewerkers met de juiste rechten hebben de mogelijkheid om de rechten van medewerker accounts aan te passen. Dit doet hij of zij door een account toe te voegen aan de gewenste rol in het medewerkersportaal.

**Scenario:** Een medewerker binnen StilteAUBv gaat voortaan werken voor de afdeling operations en heeft daarvoor andere rechten nodig. De IT medewerker wilt deze rechten toepassen in het medewerkersportaal.

**Given:** De IT medewerker heeft een bestaand account, bezit de juiste rechten, zit in het medewerkers portaal, kiest de gewenste gebruiker en drukt op de knop wijzigen.

**When:** De IT medewerker heeft op de knop wijzigen gedrukt waarna het profiel van het gekozen account getoond word.

**And:** De IT medewerker past de rol van het account aan naar operations.

**Then:** De gewijzigde gegevens zullen worden opgeslagen in de tabel `Employees`. Er zal op dat moment een update plaatsvinden op de record die van toepassing is.

**Scenario 3: Account deactiveren**

**User story:** IT medewerkers met de juiste rechten hebben de mogelijkheid om een medewerkers account te deactiveren. Dit doet hij of zij door het account op inactief te zetten in het medewerkersportaal

**Scenario:** Een medewerker binnen StilteAUBv gaat uit dienst waardoor het account gedeactiveerd moet worden. De IT medewerker wilt het account deactiveren in het medewerkersportaal.

**Given:** De IT medewerker heeft een bestaand account, bezit de juiste rechten, zit in het medewerkers portaal, kiest de gewenste gebruiker en drukt op de knop wijzigen.

**When:** De IT medewerker heeft op de knop wijzigen gedrukt waarna het profiel van het gekozen account getoond word.

**And:** De IT medewerker zit het vinkje uit bij actief.

**Then:** De gewijzigde gegevens zullen worden opgeslagen in de tabel `Employees`. Er zal op dat moment een update plaatsvinden op de record die van toepassing is.

**Scenario 4: Artikelen toevoegen**

**User story:** Een operations medewerker heeft de mogelijkheid om artikelen/producten toe te voegen aan de webshop.

**Scenario:** StilteAUBv wilt een nieuw product aanbieden op de webshop. De operations medewerker wilt dit product toevoegen aan de webshop.

**Given:** De operations medewerker heeft een bestaand account, de juiste rechten, zit in het medewerkers portaal en drukt op de knop artikelen toevoegen.

**When:** De operations medewerker heeft op de knop artikelen toevoegen gedrukt.

**And:** De site toont een formulier en de operations medewerker vult de benodigde gegevens in.

**Then:** De gegevens worden opgeslagen in de tabel `Products`. Ook word er een record aangemaakt in de tabel `ProductLogs`.

**Scenario 5: Artikelen wijzigen**

**User story:** Een operations medewerker heeft de mogelijkheid op de gegevens van een artikel/product te wijzigen in het medewerkersportaal. Denk hierbij aan het wijzigen van voorraad, prijs, naam en omschrijving.

**Scenario:** Er zijn producten geleverd waarna de voorraad op de site gewijzigd moet worden. De operations medewerker wilt dit aanpassen in het medewerkersportaal.

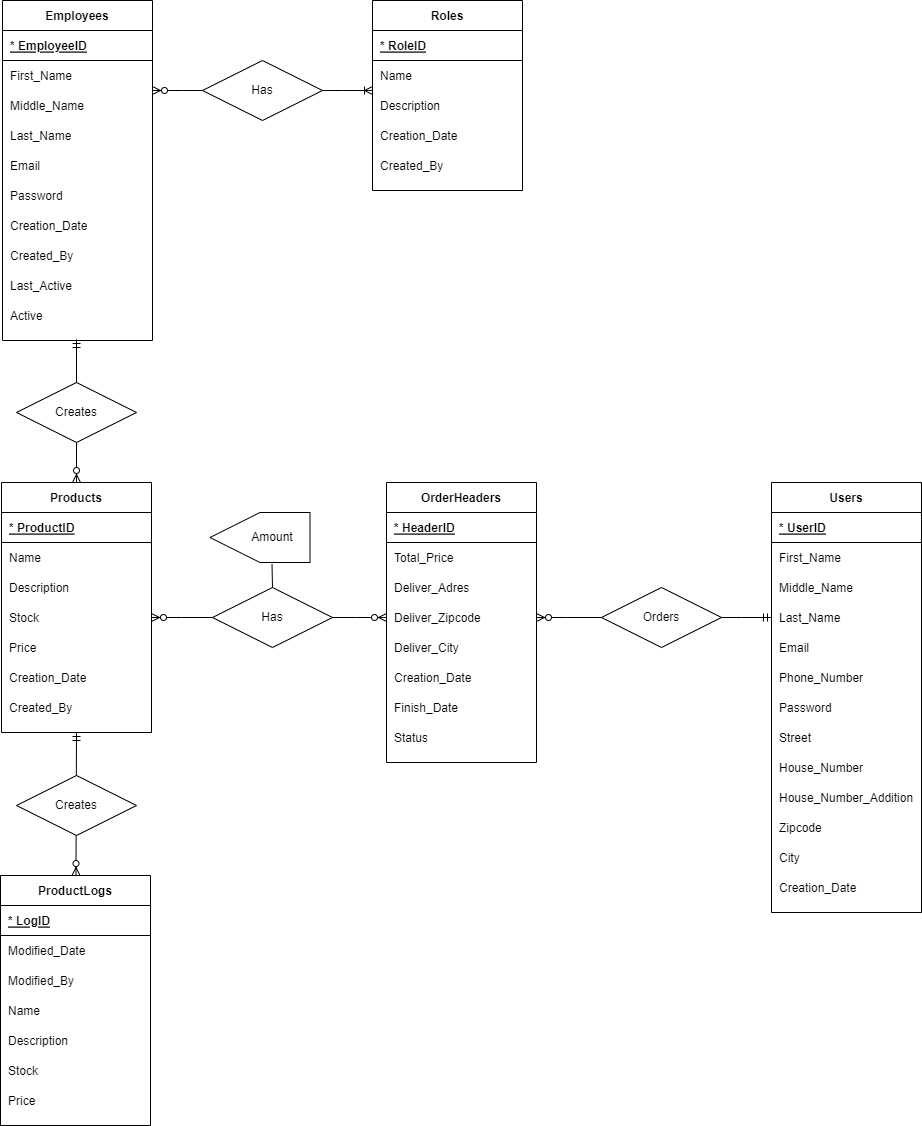
**Given:** De operations medewerker heeft een bestaand account, de juiste rechten, zit in het medewerkers portaal en drukt op de knop wijzigen naast het gewenste artikel.

**When:** De operations medewerker heeft op de knop wijzigen gedrukt.

**And:** De site toont een formulier die vooraf is ingevuld met gegevens van het gekozen artikel. Deze gegevens worden opgehaald vanuit de database en getoond in het formulier. De operations medewerker past de voorraad aan en drukt op opslaan.

**Then:** De gegevens worden opgeslagen in de tabel `Products`. Ook word er een record aangemaakt in de tabel `ProductLogs` op de wijzigingen van de producten bij te kunnen houden.

# Entity Relations Diagram (ERD)



## 4.1 Toelichting ERD

### 4.1.1 Employees

Medewerkers binnen StilteAUBv hebben de mogelijkheid om gebruik te maken van het medewerkersportaal, hiervoor hebben zij echter wel een account voor nodig. De entiteit `Employees` zullen deze gegevens opslaan en zo toegang geven tot het medewerkersportaal.

### 4.1.2 Roles

Als een medewerker een account heeft en dus bestaat in de entiteit `Employees` heeft de medewerker ook nog een rol nodig. Deze rollen worden opgeslagen in de entiteit `Roles`. Elke employee heeft tenminste 1 of meerdere rollen. Deze rollen bepalen uiteindelijk de rechten van een medewerker in het medewerkersportaal.

### 4.1.3 Products

De webshop zal uiteraard producten nodig hebben. Deze producten worden opgeslagen in de entiteit `Products`.

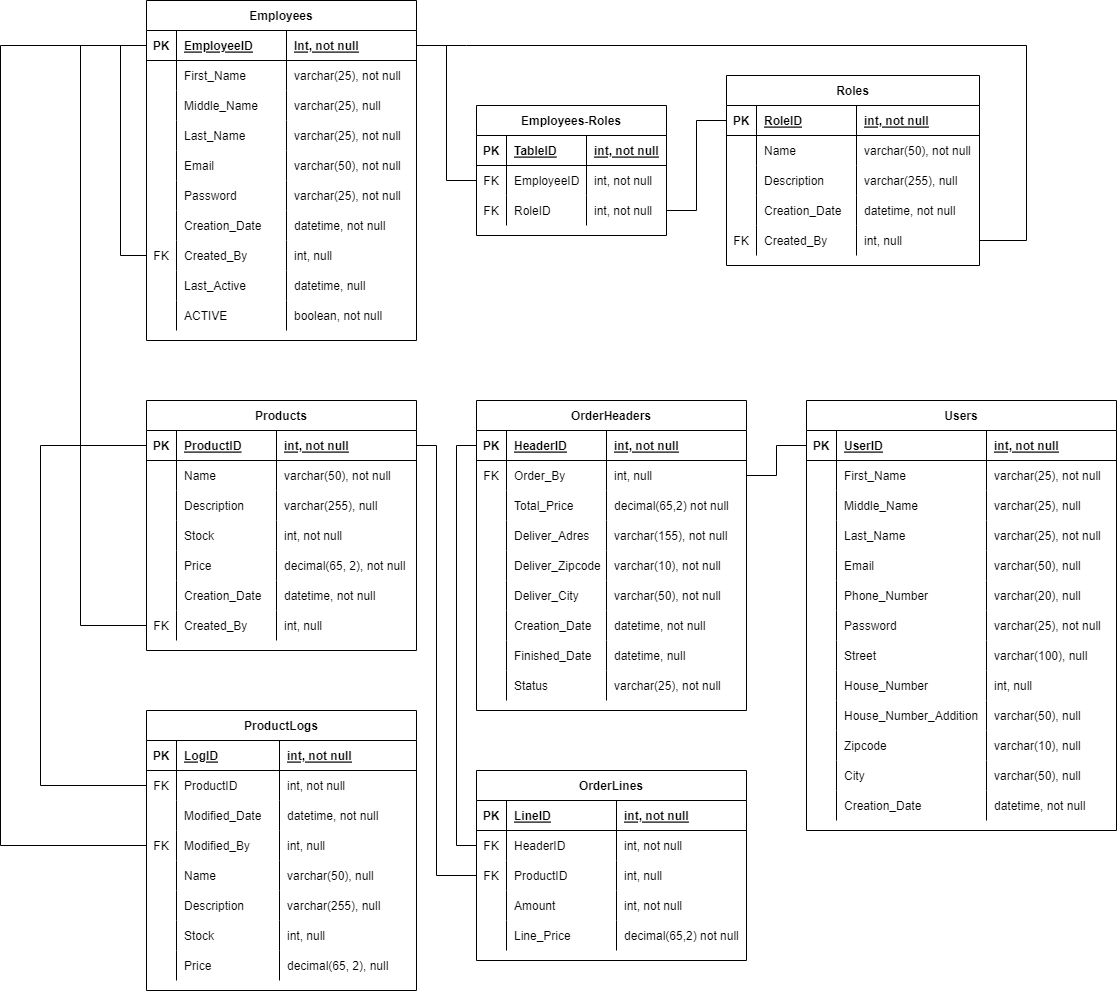
### 4.1.4 ProductLogs

Producten kunnen gewijzigd worden door de operations medewerkers van StilteAUBv. Deze wijzigingen dienen gelogd te worden voor eventuele rapportages op de instroom en uitstroom van voorraad etc. Deze wijzigingen zullen opgeslagen worden in de entiteit `ProductLogs`

### 4.1.5 Users

Klanten van StilteAUBv hebben de mogelijkheid om een account aan te maken op de website. Met dit account hebben zij toegang tot de bestellingen en statussen. Deze gegevens worden opgeslagen in de entiteit `Users`.

# Relational model (RD)



## 5.1 Toelichting RD

### 5.1.1 Employees

De medewerkers van StilteAUBv met een account staan als record in de tabel `Employees`. Elke record heeft een PRIMARY KEY `EmployeeID` en een FOREIGN KEY relatie zijn eigen table `Employees`. De FOREIGN KEY is `Created\_By`.

### 5.1.2 Roles

De rollen die gebruikt worden voor de autorisatie op de website worden opgeslagen in de tabel `Roles`. Elke record heeft een PRIMARY KEY `RoleID`. Aangezien de rollen aangemaakt worden door employees heeft de tabel `Roles` een FOREIGN KEY relatie met de tabel `Employees`. Deze FOREIGN KEY is `Created\_By` en bevat de `EmployeeID` van de employee die de rol heeft aangemaakt.

### 5.1.3 Employee-Roles

Een employee heeft tenminste 1 of meerde rollen. Om deze verbinding te kunnen maken word er gebruik gemaakt van een tussen tabel genaamd `Employees-Roles`. Elke record heeft een PRIMARY KEY genaamd `TableID` en 2 FOREIGN KEY relaties. De eerste relatie is met de table `Employees` en heet `EmployeeID`. De 2e relatie is met de tabel `Roles` en heet `RoleID`.

### 5.1.4 Products

De operations medewerkers van StilteAUBv kunnen producten aanmaken doormiddel van het medewerkersportaal. Deze producten worden opgeslagen in de tabel `Products`. Elke record heeft een PRIMARY KEY genaamd `ProductID`. Omdat de producten aangemaakt worden door employees is er een FOREIGN KEY relatie met de tabel `Employees` en de kolom heet `Created\_By`.

### 5.1.5 ProductLogs

De operations medewerkers kunnen de producten wijzigingen in de medewerkersportaal, denk hierbij aan het wijzigen van de prijs, voorraad of omschrijving. Deze wijzigingen/logging worden opgeslagen in de tabel `ProductLogs`. Elke record heeft een PRIMARY KEY genaamd `LogID`. En twee FOREIGN KEY relatie. De eerste realtie is met de tabel `Products`, de kolom heet `ProductID`. De tweede relatie is met de tabel `Employees`, de kolom heet `Modified\_By`.

### 5.1.6 Users

Klanten die een account hebben aangemaakt krijgen een record in de tabel `Users`. Elke record heeft een PRIMARY KEY genaamd `UserID` .

### 5.1.7 OrderHeaders

Klanten hebben de mogelijkheid om bestellingen te plaatsen. Deze bestellingen worden weggeschreven in twee tabellen. De samenvatting van de bestelling inclusief verzend gegevens worden opgeslagen in de tabel `OrderHeaders`. Elke record heeft een PRIMARY KEY genaamd `HeaderID` en heeft een FOREIGN KEY relatie met de tabel `Users`. Deze kolom heet `Order\_By`.

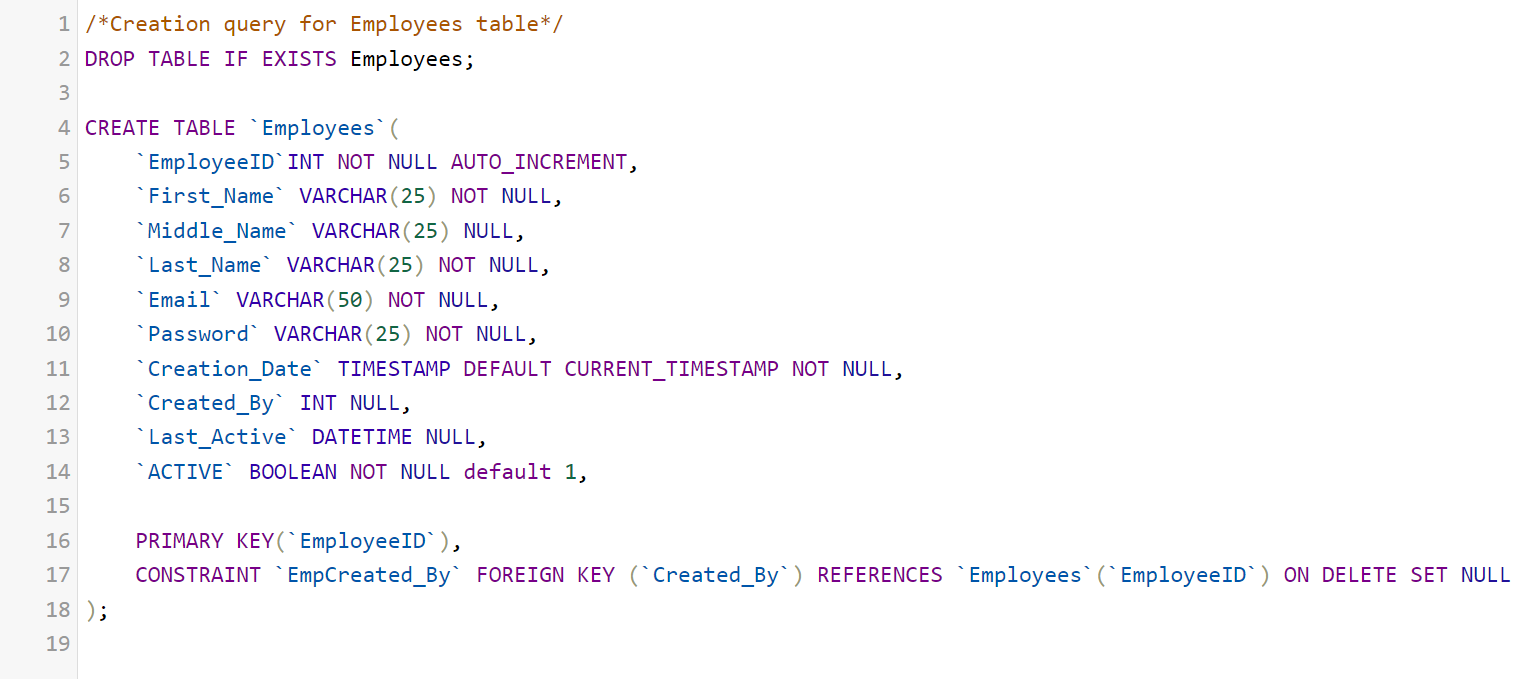
### 5.1.8 OrderLines

Zoals hierboven beschreven staat word een bestelling weggeschreven in twee tabellen. Naast de tabel `OrderHeaders` die een samenvatting is van de bestelling worden de details wegeschreven in de tabel `OrderLines`. Elke record heeft een PRIMARY KEY genaamd `LineID` en twee FOREIGN KEY relaties. De eerste relatie is met de tabel `OrderHeaders`, de kolom heet `OrderID`. De tweede relatie is met de tabel `Products`, de kolom heet `ProductID`.

# Database realisatie (DDL)

## 6.1 Tables

### 6.1.1 Structuur table Employees



In de CREATE statement voor het maken van de tabel `Employees` word er allereerst op lijn 2 gekeken of de tabel al bestaan. Indien de tabel bestaat word de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 word de CREATE statement opgebouwd. Lijn 5 creëert de kolom `EmployeeID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO\_INCREMENT.

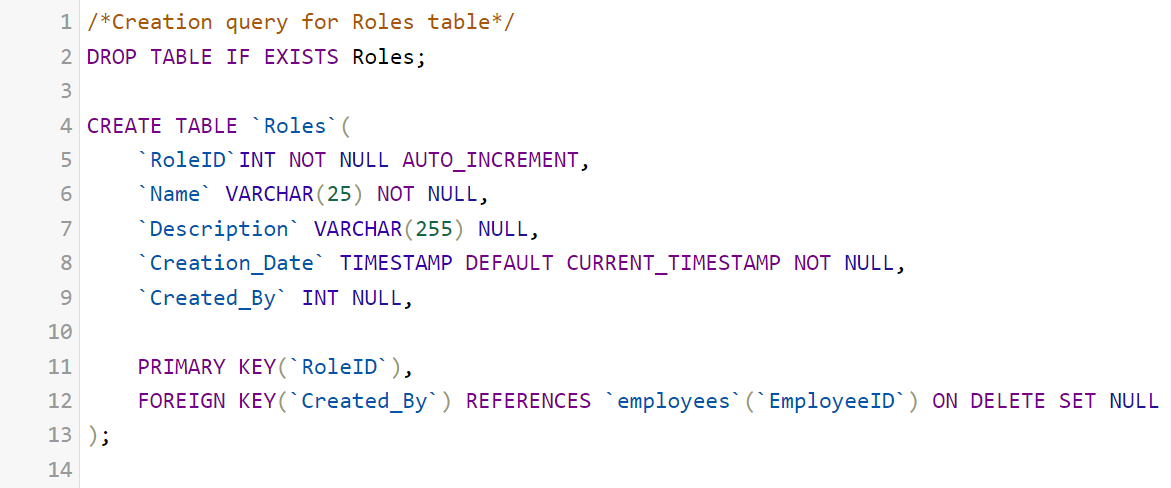
Lijn 11 creëert de kolom `Creation\_Date`. Deze heeft als standaard waarde CURRENT\_TIMESTAMP, dit zorgt ervoor dat bij elke nieuwe insert de huidige datum en tijd automatisch ingevuld word.

Als laatste is de kolom `Created\_By` die aangemaakt word op lijn 12 een FOREIGN KEY. Deze kolom refereert naar de kolom `EmployeeID` en dezelfde tabel. Deze relatie word gemaakt op rij 17. Mochten er een record verwijderd worden dan zal de waarde in `Created\_By` geüpdatet worden naar NULL.

De overige velden zijn standaard VARCHAR, INT of DATETIME en hebben verder geen uitleg nodig.



### 6.1.2 Structuur table Roles



In de CREATE statement voor het maken van de tabel `Roles` word er allereerst op lijn 2 gekeken of de tabel al bestaan. Indien de tabel bestaat word de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 word de CREATE statement opgebouwd. Lijn 5 creëert de kolom `RoleID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO\_INCREMENT.

Lijn 8 creëert de kolom `Creation\_Date`. Deze heeft als standaard waarde CURRENT\_TIMESTAMP, dit zorgt ervoor dat bij elke nieuwe insert de huidige datum en tijd automatisch ingevuld word.

Als laatste is de kolom `Created\_By` die aangemaakt word op lijn 9 een FOREIGN KEY. Deze kolom refereert naar de tabel `Employees` en daarin de kolom `EmployeeID. Deze relatie word gemaakt op rij 12. Mochten er een record verwijderd worden dan zal de waarde in `Created\_By` geüpdatet worden naar NULL.

De overige velden zijn standaard VARCHAR kolommen en hebben geen verdere uitleg nodig.



### 6.1.3 Structuur table Employees-Roles



In de CREATE statement voor het maken van de tabel `Employee-Roles` word er allereerst op lijn 2 gekeken of de tabel al bestaan. Indien de tabel bestaat word de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 word de CREATE statement opgebouwd. Lijn 5 creëert de kolom `TableID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO\_INCREMENT.

Omdat deze tabel dient als een verbinding tussen de tabellen `Employees` en `Roles` worden er 2 FOREIGN KEY relaties gelegd. Op rij 6 word de kolom `EmployeeID` aangemaakt die een FOREIGN KEY relatie heeft naar de tabel `Employees` en kolom `EmployeeID`. Op rij 7 word de kolom `RoleID` aangemaakt die een FOREIGN KEY relatie heeft naar de tabel `Roles` en kolom `RoleID`.

Voor beide FOREIGN KEY relaties geldt de regel ON DELETE CASCADE. Mochten er dus records verwijderd worden in de tabellen waar een relatie mee is dan zal de bijbehorende record ook verwijderd worden in de tabel `Employees-Roles`.



### 6.1.4 Structuur table Products



In de CREATE statement voor het maken van de tabel `Products` word er allereerst op lijn 2 gekeken of de tabel al bestaan. Indien de tabel bestaat word de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 word de CREATE statement opgebouwd. Lijn 5 creëert de kolom `ProductID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO\_INCREMENT.

Lijn 9 creëert de kolom `Price`. Deze kolom heeft als datatype DECIMAL(65,2). 65 geeft aan hoeveel cijfers er voor de komma mogen staan en de 2 geeft de hoeveelheid cijfers achter de komma aan.

Lijn 10 creëert de kolom `Creation\_Date`. Deze heeft als standaard waarde CURRENT\_TIMESTAMP, dit zorgt ervoor dat bij elke nieuwe insert de huidige datum en tijd automatisch ingevuld word.

Als laatste is de kolom `Created\_By` die aangemaakt word op lijn 11 een FOREIGN KEY. Deze kolom refereert naar de tabel `Employees` en daarin de kolom `EmployeeID. Deze relatie word gemaakt op rij 14. Mochten er een record verwijderd worden dan zal de waarde in `Created\_By` geüpdatet worden naar NULL.

De overige kolommen hebben als datatype VARCHAR of INT en hebben geen verdere uitleg nodig.



### 6.1.5 Structuur table ProductLogs



In de CREATE statement voor het maken van de tabel `ProductLogs` word er allereerst op lijn 2 gekeken of de tabel al bestaan. Indien de tabel bestaat word de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 word de CREATE statement opgebouwd. Lijn 5 creëert de kolom `LogID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO\_INCREMENT.

Lijn 6 creëert de kolom `ProductID`. Dit is een FOREIGN KEY die refereert naar de tabel `Products` en kolom `ProductID`. Deze relatie word gemaakt op rij 15. Voor deze relatie geldt de regel ON DELETE CASCADE. Mochten er dus records verwijderd worden in de tabel `Products` waar een relatie mee is dan zal de bijbehorende record ook verwijderd worden in de tabel ` ProductLogs `.

Lijn 7 creëert de kolom `Modified\_Date`. Deze heeft als standaard waarde CURRENT\_TIMESTAMP, dit zorgt ervoor dat bij elke nieuwe insert de huidige datum en tijd automatisch ingevuld word.

Als laatste is de kolom `Modified\_By` die aangemaakt word op lijn 9 een FOREIGN KEY. Deze kolom refereert naar de tabel `Employees` en daarin de kolom `EmployeeID. Deze relatie word gemaakt op rij 16. Mochten er een record verwijderd worden dan zal de waarde in `Created\_By` geüpdatet worden naar NULL.



### 6.1.6 Structuur table Users



In de CREATE statement voor het maken van de tabel `Users` word er allereerst op lijn 2 gekeken of de tabel al bestaan. Indien de tabel bestaat word de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 word de CREATE statement opgebouwd. Lijn 5 creëert de kolom `UserID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO\_INCREMENT.

Lijn 9 creëert de kolom `Creation\_Date`. Deze heeft als standaard waarde CURRENT\_TIMESTAMP, dit zorgt ervoor dat bij elke nieuwe insert de huidige datum en tijd automatisch ingevuld word.

De overige kolommen hebben de datatypes VARCHAR of INT en hebben geen verdere uitleg nodig.



### 6.1.7 Structuur table OrderHeaders

****

In de CREATE statement voor het maken van de tabel `OrderHeaders` word er allereerst op lijn 2 gekeken of de tabel al bestaan. Indien de tabel bestaat word de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 word de CREATE statement opgebouwd. Lijn 5 creëert de kolom `HeaderID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO\_INCREMENT.

Lijn 5 creëert de kolom `Order\_By` en is een FOREIGN KEY die een relatie heeft met de tabel `Users` en kolom `UserID`. Deze relatie word gemaakt op lijn 16 en heeft de regel ON DELETE SET NULL. Mocht er een record verwijderd worden uit de tabel `Users` dan zal de waarde in `Order\_By` geüpdatet worden naar NULL.

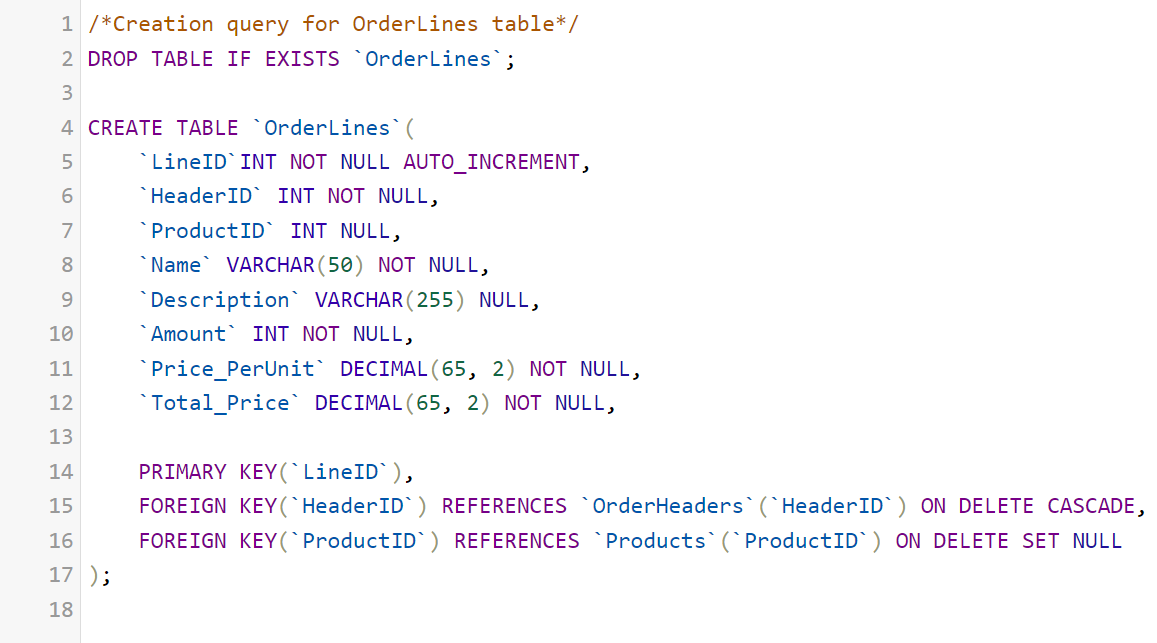
Lijn 11 creëert de kolom `Creation\_Date`. Deze heeft als standaard waarde CURRENT\_TIMESTAMP, dit zorgt ervoor dat bij elke nieuwe insert de huidige datum en tijd automatisch ingevuld word.

Lijn 12 creëert de kolom `Finished\_Date`. Deze heeft als standaard waarde CURRENT\_TIMESTAMP, dit zorgt ervoor dat bij elke nieuwe insert de huidige datum en tijd automatisch ingevuld word.

De overige kolommen hebben de standaard types VARCHAR, INT, DATETIME en DECIMAL en hebben geen verdere uitleg nodig.



### 6.1.8 Structuur table OrderLines



In de CREATE statement voor het maken van de tabel `OrderLines` word er allereerst op lijn 2 gekeken of de tabel al bestaan. Indien de tabel bestaat word de tabel verwijderd en vervolgens aangemaakt.

Vanaf lijn 4 word de CREATE statement opgebouwd. Lijn 5 creëert de kolom `LineID` wat de PRIMARY KEY zal zijn voor deze tabel. De waarde mag nooit null zijn wat te herkennen is aan NOT NULL. ook zal de ID bij elke nieuwe insert automatisch verhogen met 1. Dit is te herkennen aan AUTO\_INCREMENT.

Lijn 6 creëert de kolom `HeaderID` en is een FOREIGN KEY die een relatie heeft met de tabel `Headers` en kolom `HeaderID`. Deze relatie word gemaakt op lijn 15 en heeft de regel ON DELETE CASCADE. Mocht er een record verwijderd worden uit de tabel `Headers` dan zullen alle records met de daarbij horende HeaderID ook verwijderd worden in de tabel `OrderLines`.

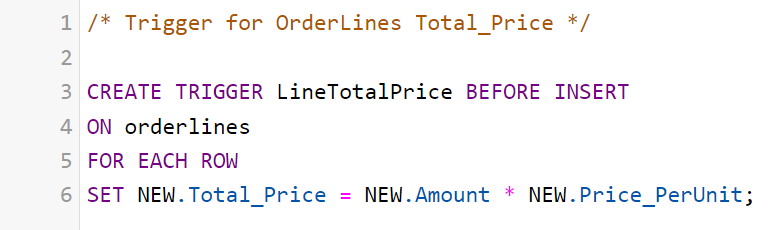
Lijn 7 creëert de kolom `ProductID` en is een FOREIGN KEY die een relatie heeft met de tabel `Products` en kolom `ProductID`. Deze relatie word gemaakt op lijn 16 en heeft als regel ON DELETE SET NULL. Mochten er records verwijderd worden uit de tabel `Products` dan zullen alle records met de daarbij horende ProductID geupdate worden met waarde null in de tabel `OrderLines`.

De overige kolommen hebben de datatypes VARCHAR, INT en DECIMAL en vergen geen verdere uitleg.



## 6.2 Triggers

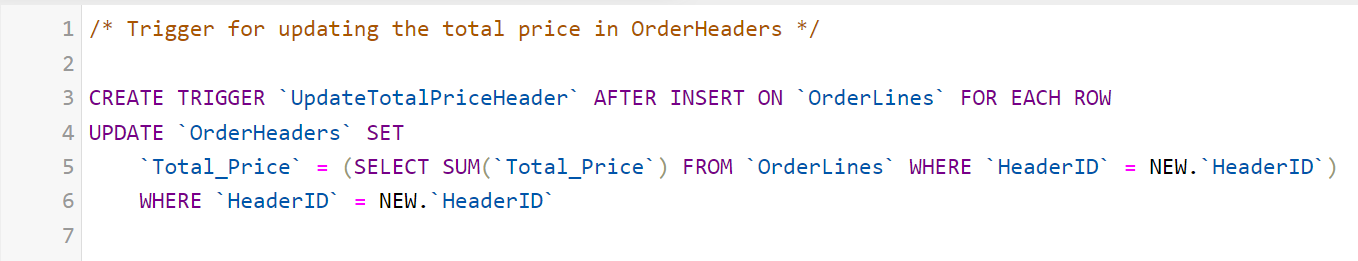
### 6.2.1 Trigger LineTotalPrice



De trigger `LineTotalPrice` zorgt ervoor dat voor elke INSERT de kolom `Total\_Price` in de tabel `OrderLines` de uitkomst krijgt van de kolommen `Amount` keer `Price\_PerUnit`. Dit zorgt ervoor dat de berekening gedaan word vanuit de database en dat deze waarde niet vanuit de website gevuld hoeft te worden. Elke regel krijgt hierdoor een totaal bedrag die weer noodzakelijk is voor het totaal bedrag van de kolom `Total\_Price` in de tabel `OrderHeaders`.



### 6.2.2 Trigger UpdateTotalPriceHeader



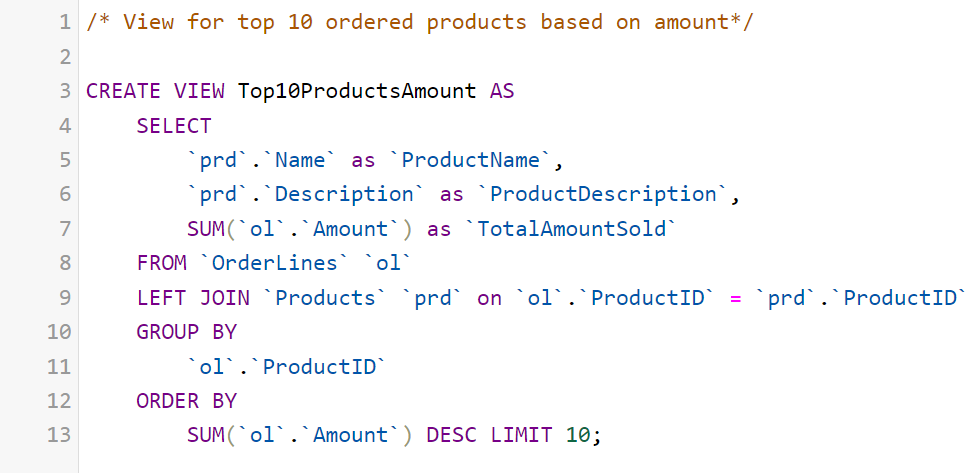
De trigger `UpdateTotalPriceHeader` zorgt ervoor dat na elke INSERT in de tabel `OrderLines` er een UPDATE plaats vindt in de tabel `OrderHeaders`. De trigger telt alle waardes van de kolom `Total\_Price` met dezelfde HeaderID van de tabel `OrderLines` bij elkaar op en update vervolgens de kolom `Total\_Price` in de tabel `OrderHeaders` met de daarbij horden HeaderID.

Dit zorgt ervoor dat de berekening automatisch word gedaan vanuit de database en dat de waarde niet vanuit de website gevuld hoeft te worden.



## 6.3 Views

### 6.3.1 View Top10ProductsAmount



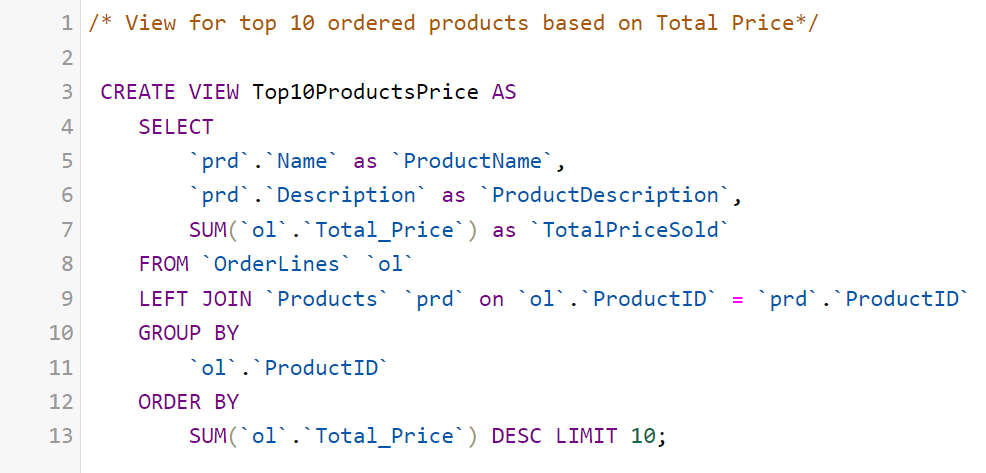
De view `Top10ProductsAmount` toont een overzicht van de bestelde producten op basis van de hoeveelheid verkochte exemplaren. In de CREATE statement word er een SUM gedaan op Amount om per ProductID de totale hoeveelheid te kunnen tonen.

Doormiddel van een LEFT JOIN word de tabel `Products` gekoppeld op basis van de `ProductID` uit `OrderLines` en de ProductID uit de tabel `Products`. Vervolgens kunnen de waardes `Name` en `Description` nu getoond worden die bij de `TotalAmountSold` horen.

Als laatste word de uitkomst gegroepeerd op `ProductID` en aflopend geordend om ervoor te zorgen dat het hoogste getal bovenaan komt te staan. Ook toont de uitkomst niet meer dan 10 records aangezien het hier gaat om de top 10.



### 6.3.2 View Top10ProductsPrice



De View `Top10ProductsPrice` word op dezelfde manier opgebouwd als de view `Top10ProductsAmount`. Het verschil is echter dat deze een SUM doet op `Total\_Price` en ook aflopen geordend word op `Total\_Price`.



### 6.3.3 View RevenuePerMonth



De view `RevenuePerMonth` toont de omzet vanuit de tabel `OrderHeaders` per jaar en maand.

Rij 5 gebruikt de functie YEAR om het jaar uit de kolom `Finished\_date` te halen.

Rij 6 gebruikt de functie CONCAT om de waarden aan elkaar toe te voegen en vervolgens in een kolom te tonen. Ook word hier de functie MONTH gebruikt om de maand als nummer uit de kolom `Finished\_date` te halen. Vervolgens word de functie MONTHNAME gebruikt om de naam van de maand uit de kolom `FInished\_date` te halen. Dit word aan elkaar geplakt doormiddel van CONCAT met een spatie tussen de twee waardes.

De reden dat de nummer van de maand voor de naam van de maand word geplakt is om deze in de juiste volgorde te tonen.

Verder word de uitkomst gegroepeerd over de uitkomst van CONCAT en de uitkomst van YEAR. Als laatste word de uitkomst op volgorde gezet doormiddel van de uitkomst van YEAR en de uitkomst van CONCAT.



# Security (DCL)

## 7.1 Admin

Voor de toegang van de database zelf zal er een admin account aangemaakt worden die volledige rechten heeft om alle tabellen.





De user `admin` word aangemaakt inclusief wachtwoord op lijn 1. Achter `admin`@ staat een % teken, dit betekent dat de user een connectie kan maken vanaf elke host. Als voorbeeld had hier ook localhost kunnen staan wat in dat geval zou betekenen dat de user alleen had kunnen inloggen via de server zelf.

Op lijn 2 word aangegeven dat de user `admin` volledige rechten heeft op de gehelde database `stilteaubv`.

## 7.2 system

Vanuit de site zullen de SELECT, INSERT, UPDATE, en DELETE statements plaatsvinden. Voor de site zal een aparte user aangemaakt worden die deze functies mogen gebruiken op alleen de benodigde tabellen.





De user `system` word aangemaakt inclusief wachtwoord op lijn 1. Ook hier staat een % teken wat ervoor zorgt dat de user een connectie kan maken vanaf elke host.

Op lijn 2 worden de rechten SELECT, INSERT, UPDATE, en DELETE gegeven op alle tabellen in stilteaubv.

# Maintenance

De database wordt onderhouden via phpMyAdmin. Er zijn een aantal functies binnen phpMyAdmin die gebruikt kunnen worden voor het onderhouden van de database. Zie hieronder de instructies voor deze functies:

## 8.1 Instructie backup

1. Start **phpMyAdmin**.
2. Selecteer de database `**stilteaubv**`.
3. Druk op de knop **Export**.
4. Druk op de knop **Go**.
5. Het SQL bestand word gedownload en staat in je download folder. Verplaats dit bestand naar de backup server.

## 8.2 Instructie analyze table

1. Start **phpMyAdmin**.
2. Druk op het **plusje** links van de database `**stilteaubv**`.
3. Druk op de knop **Tables**.
4. Druk op de knop **Check all**.
5. Druk op de dropdown **With selected:**
6. Selecteer de optie **analyze** **table**.
7. De analyse loopt nu en geeft de status per tabel aan.

## 8.3 Instructie Check table

1. Start **phpMyAdmin**.
2. Druk op het **plusje** links van de database `**stilteaubv**`.
3. Druk op de knop **Tables**.
4. Druk op de knop **Check all**.
5. Druk op de dropdown **With selected:**
6. Selecteer de optie **check** **table**.
7. De analyse loopt nu en geeft de status per tabel aan.

## 8.3 Instructie optimize table

1. Start **phpMyAdmin**.
2. Druk op het **plusje** links van de database `**stilteaubv**`.
3. Druk op de knop **Tables**.
4. Druk op de knop **Check all**.
5. Druk op de dropdown **With selected:**
6. Selecteer de optie **optimize** **table**.
7. De analyse loopt nu en geeft de status per tabel aan.

# Installatie

## 9.2 Installatie handleiding

## 9.3 Insert data

### 9.3.1 Insert Employees





### 9.3.2 Insert Users



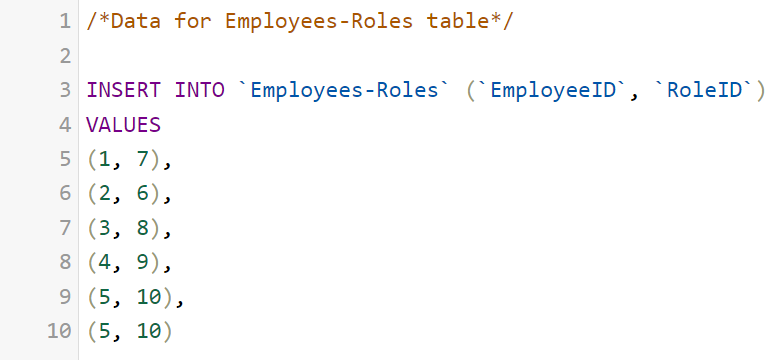


### 9.3.3 Insert Roles



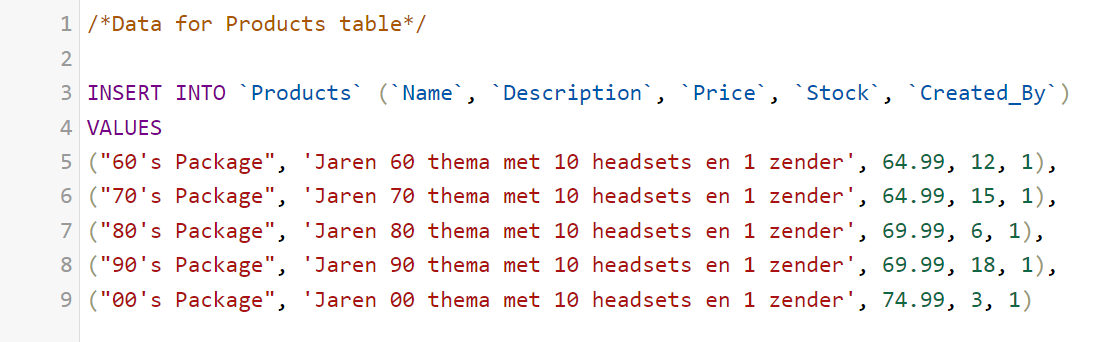


### 9.3.4 Insert Employees-Roles



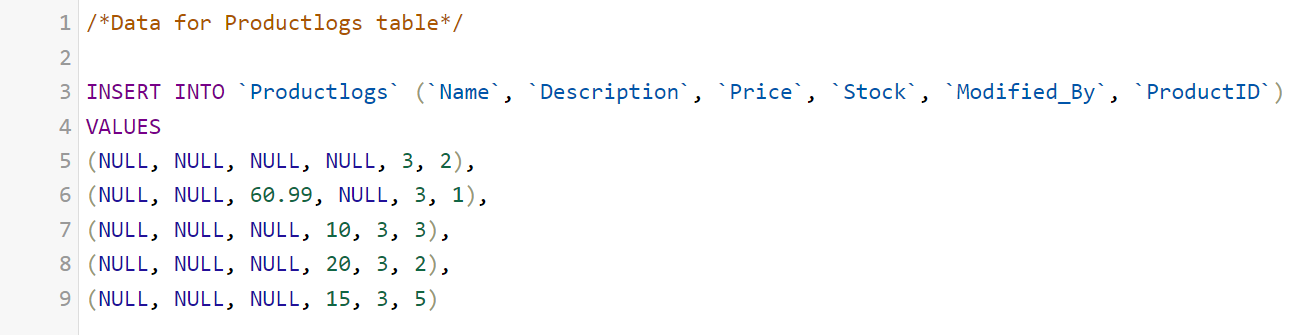


### 9.3.5 Insert Products



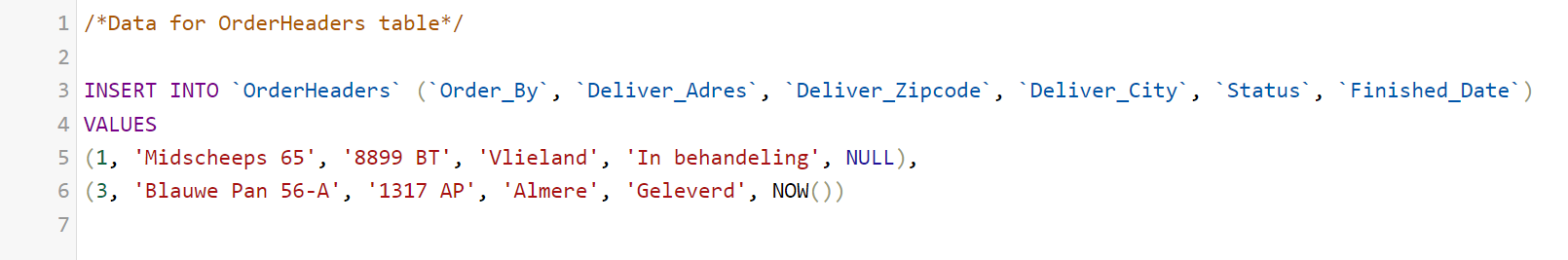


### 9.3.6 Insert ProductLogs



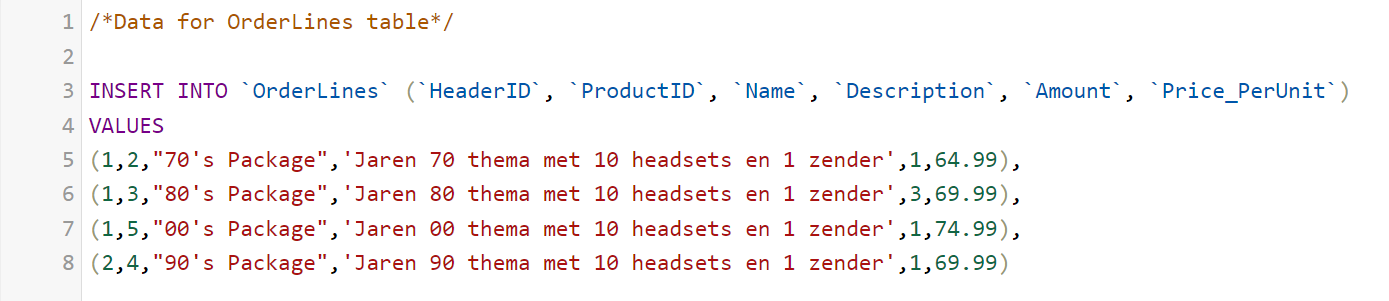


### 9.3.7 Insert OrderHeaders





### 9.3.8 Insert OrderLines





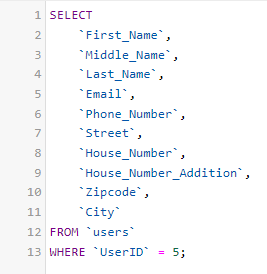
# 10 Uitwerking (DML)

De site zal verschillende scenario’s moeten kunnen uitvoeren op de database. Deze scenario’s zijn hieronder uitgeschreven.

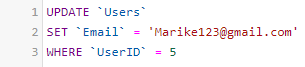
## 10.1 Edit user

De klant moet de mogelijkheid hebben om zijn of haar gegevens in te zien en deze te wijzigen. In dit voorbeeld word er gebruik gemaakt van de user Marike Vlieland met UserID 5 en wijzigen we het email adres van Marike.

In de onderstaande query word de SELECT statement gebruikt om de data van Marike op te halen. Om te bepalen dat het om Marike gaat word er gebruik gemaakt van WHERE `UserID` = 5.

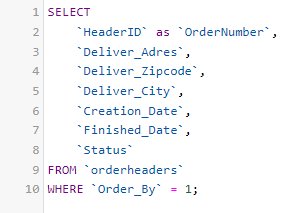


In de onderstaande query heeft Marike bepaalt dat het email adres gewijzigd moet worden. Dit zal voor een UPDATE statement zorgen waarin het email adres gewijzigd word en de user bepaald word doormiddel van WHERE `UserID` = 5.



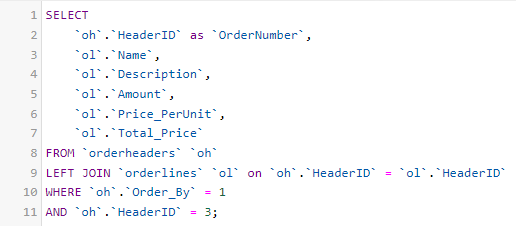
## 10.2 Show orders

De klant heeft in het klantenportaal ook de mogelijkheid om zijn of haar bestellingen in te zien. Voor dit voorbeeld gebruiken we de user Hans met UserID 1. In de onderstaande query word er gebruik gemaakt van de SELECT statement om zo alle orders te tonen van Hans.



## 10.3 Show order details

In hoofdstuk 10.2 Show orders worden de bestellingen van Hans getoond op de tabel `OrderHeaders`. Dit levert als resultaat het bestelnummer, aflevergegevens en de status op. De gebruiker moet ook de mogelijkheid hebben om de details op regel niveau te bekijken. In de onderstaande query word er gebruik gemaakt van de SELECT statement om zo de details de tonen van de bestelling met HeaderID 3. Om de details te tonen word er gebruik gemaakt van LEFT JOIN om de details uit de tabel `OrderLines` te halen.



## 10.4 Edit products

Een operations medewerkers moet de mogelijkheid hebben om producten te wijzigen. In dit voorbeeld wijzigen we de voorraad en prijs van het product 90’s Package met ProductID 4. De wijzigingen zullen doorgevoerd worden door de user Bart met EmployeeID 3.

In onderstaande query word er gebruik gemaakt van de UPDATE statement waarin de voorraad en prijs gewijzigd word. Ook zorgt het wijzigen van een product voor een INSERT op de tabel `ProductLogs`.

