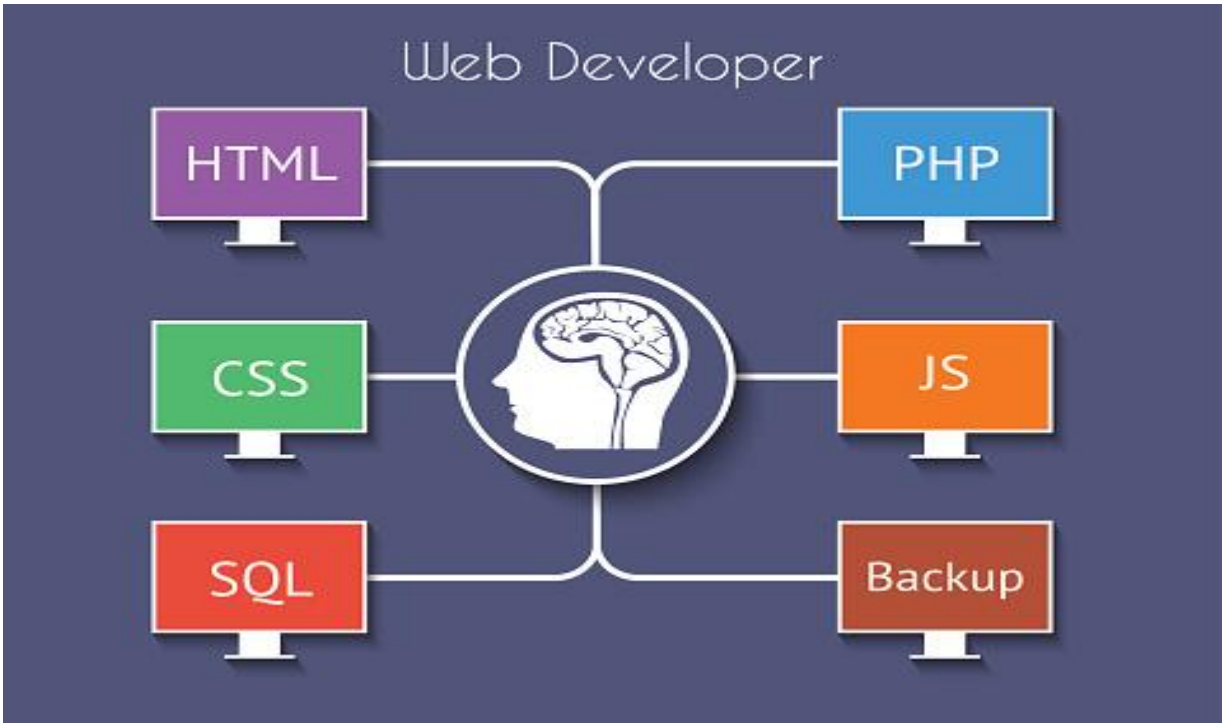
Minna, Salam & Jan, Erhvervsakademi Kolding

PBI - Projekt, uge 8, 2022



17.-24. februar 2022

Indledning

I dette delprojekt, vil der som udgangspunkt blive løst et opgavesæt med redskaber og værktøjer, som vi har lært igennem undervisningen i dataforståelse, databasedesign og webteknologi.

I dataforståelse og databasedesign er vi teknisk set kommet hele vejen rundt i brugen af SQL til oprettelse af relationsdatabaser, administration af SQL-server, brugeroprettelse, brugerrettigheder, diagramtegning såsom f.eks. ER-modeller, og sidst men ikke mindst, at kunne lave korrekte SQL-forespørgsler.

I webteknologi, har vi indtil videre arbejdet med HTML og CSS, og disse to redskaber skal vi bruge til at skabe et website til at demonstrere vores indlæring i HTML, CSS, og samtidigt skal svarene på opgaverne indgå som indhold på det website, der skal skabes.

Databasedelen omhandler oprettelse af en relationsdatabase, som danner udgangspunkt for besvarelse af de fleste af spørgsmålene i opgavesættet. Vi skal bl.a. skabe et ordresystem i SQL, hvor man via kald af en stored procedure, skaber en ny ordre med tilhørende ordrelinje og med en ny bruger, der bliver lagret i en database, og så skal vi oprette en XML-fil med en datastruktur magen til vores ordretabel, og et XML schema, hvor man bl.a. beskriver datatyperne af indholdet i XML-filen. XML schema har i øvrigt filendelsen, xsd, og ikke xml.

For at folk ikke skal blive forvirret over ord som entity og attribute, så må vi hellere lige forklare, at i fagsprogets verden indenfor databaser, der hedder en tabel en entity, og en kolonne hedder en attribute, og en række hedder en tuple.

Database design

Database Normalisering er processen med at indsamle egenskaber i form af ordnede relationer og transformere data i form af binære tabeller, og dette kaldes første normale formular.

1NF:

I det Emne vi finder mange tomte værdi i nogle kolonne i 1FN . som i order Nr, order Date og customer type. så vi bare skal bryde værdierne ind i atom værdier. Den 1Nf heller ikke multe value attributes. Den duplikerede data skal sammensættes hvis de er relaterede.så lægger vi resten af dataene i anden tabel.
Ved hjælp af formularen Første normale tilstand øges dataredundansen, da der vil være mange kolonner med de samme data i flere rækker, men hver række som helhed vil være entydig.

2NF:

til at blev 2NF ,så tabellen skal være i den første normale form.Den anden fase, vi skal se sammenhængene, er de i den første tabel er totalt afhængige eller delvist.
Den anden normale form skal ikke delves afhængighed.hvor en attribut i en tabel kun afhænger af en del af den primære nøgle og ikke af hele nøglen
så i tabel er den primære nøgle vil være unik for hver række. de skal vær total afhængighed.Såsom række data fra denne tabel products.
så product-id er PK ,og product navn , antal produkter, unit price afhang af product-id. så vi er i 2FN.

3NF:

En tabel er i tredje normalform, hvis den opfylder kravene til anden normalform, ingen af dens ikke-nøglefelter er funktionelt afhængige af noget andet ikke-nøglefelt.
for eksempel i tabel orderline ,For at løse dette problem er vi nødt til at fjerne den transitive afhængighed, der kan gøres ved at oprette en anden tabel, f.eks.orderline, der indeholder 5 kolonner indeholder (orderline-id, order-id, product-id,antal og price) dvs. orderline-id (som primær nøgle) og order-id , product -id er en fremmednøgle til orders, products tabellen .

BCNF:Boyce-codd Normal form,er en forlængelse af tredje normal form.

BCNF skal kon voliere disse emner for bedre forståelse af databasen.

Hvis forholdet mellem kandidatnøgler ikke er klart, skal adskille disse nøgler i forskellige tabeller.

Database:

Som en del af projektet, har vi designet en relationsdatabase ud fra de oplysninger og præmisser, der er angivet i opgavesættet, og den består af fire entities(tabeller), som hver især indeholder et antal attributter(kolonner).

Projektets database, har vi kaldt Project-store, og den simulerer, hvordan man f.eks. kan oprette en ordre i en webshop.

Nedenstående SQL-kode opretter databasen med to foruddefinerede ordrer, to foruddefinerede stored procedurer, og tre triggere.

```
DROP DATABASE IF EXISTS Project_store;
CREATE DATABASE Project_store;
USE Project_store;

CREATE TABLE Customers(
Customer_ID int NOT NULL AUTO_INCREMENT,
CustomerType varchar(45) NOT NULL,
PRIMARY KEY(Customer_ID)
);
insert into Customers(CustomerType) values('Wholesaler');
insert into Customers(CustomerType) values('Retailer');

CREATE TABLE Orders(
Order_ID varchar(45) NOT NULL,
Customer_ID int NOT NULL,
Order_date date,
PRIMARY KEY(Order_ID),
FOREIGN KEY (Customer_ID) REFERENCES Customers(Customer_ID)
);
insert into Orders(Order_ID, Customer_ID, Order_date)
values('20210226001', 1, '2021-02-26');
insert into Orders(Order_ID, Customer_ID, Order_date)
values('20210226002', 2, '2021-02-26');

CREATE TABLE Products(
Product_ID int NOT NULL AUTO_INCREMENT,
Name varchar(45) NOT NULL,
Quantity_in_stock int NOT NULL,
Unit_price float NOT NULL,
PRIMARY KEY(Product_ID)
);
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item1', 20, 2);
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item2', 10, 112);
```

```
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item3', 20, 2);
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item4', 3, 2);
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item5', 15, 0.2);
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item6', 20, 2);
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item7', 20, 2);
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item8', 20, 2);
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item9', 20, 2);
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item10', 50, 0.15);
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item11', 20, 2);
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item12', 20, 2);
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item13', 20, 2);
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item14', 150, 0.45);
insert into Products(Name, Quantity_in_stock, Unit_price)
values('Item15', 20, 2);

CREATE TABLE Orderline(
Orderline_ID int NOT NULL AUTO_INCREMENT,
Order_ID varchar(45) NOT NULL,
Product_ID int NOT NULL,
Quantity int NOT NULL,
Price float NOT NULL,
PRIMARY KEY(Orderline_ID),
FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID),
FOREIGN KEY (Product_ID) REFERENCES Products(Product_ID)
);
insert into Orderline(Order_ID, Product_ID, Quantity, Price)
values('20210226001', 10, 150, 22);
insert into Orderline(Order_ID, Product_ID, Quantity, Price)
values('20210226001', 14, 100, 45);
insert into Orderline(Order_ID, Product_ID, Quantity, Price)
values('20210226002', 2, 5, 560);
insert into Orderline(Order_ID, Product_ID, Quantity, Price)
values('20210226002', 5, 8, 40);
insert into Orderline(Order_ID, Product_ID, Quantity, Price)
```

```
values('20210226002', 10, 1, 25);
```

Stored procedure:

I tillæg til ovenstående kode, følger disse to stored procedurer, som bliver oprettet med nedenstående kode.

Den første procedurer kan bl.a. udregne totalprisen på en ordre med flere enheder af den samme vare, og den anden procedure viser en foruddefineret fejlmeddelelse, når antallet af enheder af en bestemt vare er under fem i lagerbeholdningen.

```
DELIMITER //
CREATE PROCEDURE `CreateOrder`(in cCustomertype varchar(45), in
cProduct_ID int, in cQuantity int)
cr: BEGIN
  DECLARE cdate int;
  DECLARE max_value INT;
  DECLARE cOrder_ID varchar(45);
  DECLARE cCustomer_ID int;
  DECLARE cUnit_price float;
  DECLARE cPrice float;
  DECLARE cQuantity_in_stock int;

  select Quantity_in_stock into cQuantity_in_stock from Products where
Product_ID = cProduct_ID;

  if cQuantity_in_stock < 1 or cQuantity > cQuantity_in_stock then
     LEAVE cr;
  end if;

  SET cdate=CURRENT_DATE();
```

```
  select right(max(Order_ID), 3) into max_value FROM orders;
  set max_value = max_value + 1;
  if max_value < 10 then
      set cOrder_ID = concat(cdate, '00', max_value);
  elseif  max_value > 9 and max_value < 100 then
    set cOrder_ID = concat(cdate, '0', max_value);
  elseif  max_value > 99 and max_value < 1000 then
    set cOrder_ID = concat(cdate, max_value);
  end if;
```

```
  insert into Customers(CustomerType) values(cCustomertype);
  select max(Customer_ID) into cCustomer_ID from customers;
  insert into Orders(Order_ID,  Customer_ID, Order_date)
values(cOrder_ID, cCustomer_ID, cdate);
  select Unit_price into cUnit_price from Products where Product_ID =
cProduct_ID;
  set cPrice = cUnit_price * cQuantity;
  insert into Orderline(Order_ID, Product_ID, Quantity, Price)
values(cOrder_ID, cProduct_ID, cQuantity, cPrice);
  update Products set Quantity_in_stock = Quantity_in_stock - cQuantity
where Product_ID = cProduct_ID;
END //

DELIMITER ;

DELIMITER //
CREATE PROCEDURE `CheckQuantityInStock`(in cProduct_ID int)
BEGIN
  DECLARE cQuantityInStock int;
  select Quantity_in_stock into cQuantityInStock from Products where
Product_ID = cProduct_ID;
  if cQuantityInStock < 5 then
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Mindre end fem enheder på lager af den
pågældende vare!';
  end if;
END //

DELIMITER ;
```

Trigger:

En trigger er en mekanisme, der har til formål at udføre en handling, enten før eller efter den første handling, før man kan udløse en trigger, så skal den defineres først, og samtidigt skal det definerer, hvad der udløser en trigger.

I vores eksempel har vi defineret tre triggere, der kan udløses før en anden handling, og den handling, der udløser triggeren skal i øvrigt også defineres. I de tre triggere herunder, der skaber man først triggeren, og samtidigt definerer man, hvad der skal udløse den.

Som man kan se i koden til den første trigger, så bliver den kun udløst, hvis man laver en UPDATE på order-tabellen, og det triggeren gør, er, at den kopierer det man vil opdatere over i en anden tabel, inden den egentlige opdate bliver udført. Derved har man lavet en slags backup af de data, som man ville opdatere.

```
CREATE TRIGGER before_order_update
BEFORE UPDATE ON Orders
FOR EACH ROW
INSERT INTO Orders_audit
SET action = 'update', Order_ID=OLD.Order_ID,
Customer_ID=OLD.Customer_ID, Order_date = OLD.Order_date, changedat =
NOW();

CREATE TRIGGER before_order_delete
BEFORE DELETE ON Orders
FOR EACH ROW
INSERT INTO Orders_audit
SET action = 'delete', Order_ID=OLD.Order_ID,
Customer_ID=OLD.Customer_ID, Order_date = OLD.Order_date, changedat =
NOW();

CREATE TRIGGER before_order_insert
BEFORE INSERT ON Orders
FOR EACH ROW
INSERT INTO Orders_audit
SET action = 'insert', Order_ID=NEW.Order_ID,
Customer_ID=NEW.Customer_ID, Order_date = NEW.Order_date, changedat =
NOW();
```

Transaction:

En transaction er en mekanisme, som man kan bruge i situationer, hvor det er vigtigt, at f.eks. hvis en procedure pludseligt bliver afbrudt midt i det hele og kun er blever udført halvt, måske på grund af et nedbrud, så kan man indkapsle den i en transaction og commit, og det betyder, at hvis der opstår et nedbrud under udførelsen af proceduren, så bliver udførelsen rullet tilbage, så den ikke kun er udført halvt, og derved kan proceduren starte forfra med at blive udført korrekt, når serveren er oppe at køre igen.

Brugen af transaction foregår på følgende måde:

```
START TRANSACTION;
call CreateOrder("Testsaler", 12, 4);
COMMIT;
```

Brugere:

Oprettelse af brugere i SQL foregår på følgende måde, og som man ser i nedenstående session, så kan man oprette flere brugere på én gang.

```
drop user if exists alice@localhost, bob@localhost, super@localhost;
create user if not exists bob@localhost identified by 'test',
alice@localhost identified by 'test', super@localhost identified by
'test';
```

Privileges:

Når en eller flere brugere er oprettet, så skal de tildeles nogle rettigheder, og de rettigheder sættes på følgende måde. F.eks. så ser vi, at to af brugerne får tildelt INSERT, OPDATE og DELETE rettigheder på på alle databaser og tabeller, og en tredje bruger der kun får tildelt SELECT rettigheder.

Man kan også definere rettigheder, der kun gælder en bestemt database eller entities(tabeller) i en database.

Som vi ser i den første GRANT, så kan man også her tildele flere forskellige rettigheder til flere forskellige brugere på én gang

```
GRANT INSERT, UPDATE, DELETE ON *.* TO bob@localhost, alice@localhost;
GRANT SELECT ON *.* TO super@localhost;
```

Det viser sig faktisk, at selv om Bob og Alice har insert, update og delete rettigheder, så har de i virkeligheden kun insert rettigheder, for det kræver faktisk select rettigheder, at udføre update og delete.

```
insert into Customers(CustomerType) values('Bob');
insert into Customers(CustomerType) values('Alice');
```

Super kan til gengæld udføre select statements, men har ikke rettigheder til insert, update og delete.

```
select Orders.Order_ID, Orderline.Product_ID, Order_date, CustomerType,
Quantity, Price from Orders
join Customers on Customers.Customer_ID=Orders.Customer_ID
join Orderline on Orderline.Order_ID=Orders.Order_ID
join Products on Products.Product_ID=Orderline.Product_ID;
```

XML:

XML står Extensible Markup Language, og er et opmærkningssprog, hvor man selv definerer sine tags, og det bruges primært til udveksling af information mellem flere computere.

I tillæg til en XML-fil, kan man have et XML schema, som indeholder de korrekte datatyper for indholdet af XML-filen. En XML-fil har xml som filendelse, og et XML schema har xsd som filendelse.

Nedenstående to eksempler er en XML-fil som indeholder vores ordre-tabel, og et XML schema som indeholder datatyperne for indholdet i XML-filen.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Orders>
  <Order>
    <Order_ID>20210226001</Order_ID>
    <Customer_ID>1</Customer_ID>
    <Order_date>2021-02-26</Order_date>
  </Order>
  <Order>
    <Order_ID>20210226003</Order_ID>
    <Customer_ID>3</Customer_ID>
    <Order_date>2022-02-21</Order_date>
  </Order>
  <Order>
    <Order_ID>20210226004</Order_ID>
    <Customer_ID>4</Customer_ID>
    <Order_date>2022-02-21</Order_date>
  </Order>
</Orders>
```

I tillæg til XML-filen, er der også et XML schema, og det ser sådan ud:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Orders">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Order_ID" type="xs:string"/>
      <xs:element name="Customer_ID" type="xs:integer"/>
      <xs:element name="Order_date" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Webteknologi

HTML5 er et programmeringssprog, som står for Hyper Text Markup Language. Det er et system, der gør det muligt at ændre udseendet af websider, samt foretage justeringer af deres udseende. Det plejede også at strukturere og præsentere indhold til nettet.
Med HTML5 kan browsere som Chrome, Firefox, Safari osv., hvordan man viser en bestemt webside, vide, hvor elementerne er, hvor billederne skal placeres, og hvor teksten skal placeres. Dette bruges til dette projekt, der laves en webside hvoraf der vil bruges html og css af dele af rapporten.

Der blev startet med at planlægge strukturen på hvordan man gerne vil have med i den. Der blev benyttet af det der blev lært fra lektionerne til at lave en html side.
Der laves en forside/index i en html fil hvoraf den skal indeholde en menubar for at finde rundt i alle opgaverne. Menubaren bliver lavet ved at bruge unordered list (ul) hvor resterende bliver tilpasset i css dokumentet ved at inddele i classes..
Class er en attribut, som angiver et eller flere navne for et HTML-element. Class-attributten kan bruges på ethvert HTML-element og kan bruges af CSS og JavaScript til at udføre visse opgaver for elementer med det angivne class navn. Ved det kan der angives farver, skrifttype, størrelse, layout osv. til html filen ved hjælp af stylesheet (CSS filen).

Konklusion:

Arbejdet med dette delprojekt, har på mange måder været lærerigt og udfordrende, og vi har i gruppen fået prøvet nogle ting af, som man normalt ikke lige kaster sig ud i, med mindre man har et projekt, som kræver at man bruger de redskaber, der f.eks. findes i SQL, HTML og CSS.

Fordelingen af arbejdsopgaver har vi også været gode til at fastsætte i forhold til, hvem der havde lyst til at påtage sig de enkelte delopgaver.

I det hele taget, er det meget lærerigt, hver gang vi har et projekt, og man ser faktisk frem til det næste projekt, i forhold til, hvad det byder på af nye udfordringer, sammenholdt med den indlæring vi har fået i den mellemliggende periode.