Anslag: 21284

Multimedia Design: Semester 3 - projekt 01

Projekt: Database

Projektperiode: 07. September – 20. September 2015

SQL

Gruppe nummer: MulB07

Vejledere: Ivan Rosenvinge Frederiksen & Tuje Becher

MULA

Anastasia Keller
cph-ak186@cphbusiness.dk

MULB

Benjamin Lipsø
cph-bl91@cphbusiness.dk

René Abel Jensen
cph-rj135@cphbusiness.dk

Sabine Larsen
cph-sl176@cphbusiness.dk

Indholdsfortegnelse:

1. Opgavebeskrivelse

1.1 Problemformulering

For tiden er der en Skolestart kampagne Apple holder for at tiltrække flere studerende til at købe Apple produkter. Der tilbydes hovedtelefoner gratis, hvis man køber en Mac bærbar, en iMac (stor-skærm) eller en iPad.
Derfor har vi i gruppen har valgt Apple som den e-shop vi gerne vil tilbyde en database løsning.
Apples hjemmeside http://www.apple.com/dk er god nok i forhold til kunde-side. Man kan godt få oversigt over flere produkter og sætte dem ind i shopping "bag" (som står for shopping "cart"). Det er også muligt at få sin ordre sporet, tjekke ordrens status, ændre ordren, returnere produkt(er), eller slette den.
Vi vil skabe en separat database for nogle af Apples produkter tilegnet specifikt til den her kampagne. Databasen ville kunne bruges til at lave en hjemmeside følgelig til studerende og skoler, hvor de kan vælge kun de produkter som er en del af Apples Skolestart kampagne. [1]

1.2 Projekt planlægning

Efter det blev klart hvad vi skal lave i dette projekt, gik vi i gang med planlægningen. Først lavede vi en oversigt over alle dele af opgaven som skal laves i form af PBS, der ligger i Appendix 1. Efterfølgende lavede vi et Gantt-chart og WBS, som kan ses i Appendix 2 og Appendix 3. Gantt-charten viser alle trin i projektet og hvor lang tid det tog at udføre dem og hvem i gruppen der er ansvarlig for hvilken opgave. WBS giver oversigt over stort set det samme, bare mere detaljeret.

2. User Story

2.1 User story

En user story bliver brugt erhverv mæssigt til udvikling og forbedring.
Når man skriver en user story er det vigtig at man beskriver hvem brugeren er, hvad bruger gerne vil opnå og hvad motivere bruger.
I dette tilfælde er "bruger" vores kunde, og derfor er det til vores store interesse at kunden får dækket så meget af sine behov som muligt så de har lyst til at købe deres vare hos firmaet.

Vores user stories er skrevet efter en model, som har hjulpet os med at sikre at vi får de vigtigste informationer med, som bl.a. tekniske krav, begrænsninger og forudsætninger.
modellen vi har brugt er skrevet således: "Som en___, vil jeg gerne, for at____". '
Som et erhverv vil jeg gerne kunne logge ind når købe flere vare ad gangen, så firmaet kan huske hvilke modeller jeg købte sidste gang jeg skal handlede.

- Som studerende vil jeg gerne have anbefalet de bedste Apple vare for mig, så min skolestart kan starte uden bekymringer
- Som kunde vil jeg gerne kunne se og bestille produkterne i flere farver, da jeg godt kan lide at mine gadgets har samme farve.
- Som kunde vil jeg gerne kunne bestille en computer med højest hukommelse, da jeg bruger min computer til videoredigering.
- Som erhverv vil jeg gerne have svar på mine forespørgsler, så jeg føler mig sikker i mit køb.

Da vi har koncentreret os om unge og skolestart har vi valgt at tage udgangspunkt i de vare der ville passe bedst til vores målgruppe, så de ikke skal bruge længere tid end nødvendigt når de skal købe deres vare. Det har selvfølgelig givet kunderne et begrænset udvalg, men med user stories kan man tage udgangspunkt i om vi har fanget målgruppen og deres søgen på bl.a. en god skole computer.

2. User Story

2.2 Detaljeret User stories

Navn:
Scenarie 1

Identifier:
US01

Forudsætninger:
Forbindelse til databasen ved login
Forbindelse til support

Grundkursus:
Use case begynder kunden logger ind med en bruger og hans køb bliver registreret inde i databasen. Når kunden skal vide hvilke modeller han
købte sidste gang han handlede, kan han skrive til support og de kan printe en liste ud fra databasen.
User story slutter når kunden har fået svar fra supporten.

Alternativ Kursus A:
Et alternativ ville være at kunden selv kunne se sine tidligere ordre når logger ind på deres bruger.

Navn:
Scenarie 3

Identifier:
US02

Forudsætninger:
Forbindelse til databasen på hjemmesiden

Grundkursus:
User story begynder når kunden leder efter flere farver på sine gadgets.
På hjemmesiden kunne man sortere produkterne efter farve så vores kunde kun kan se de relevante vare. Disse oplysninger bliver trukket via databasen som gemmer disse informationer.
User story slutter når kunden har fået sorteret alle produkterne i farve kategori.

Alternativ Kursus A:
Alternativ ville være at kunden individuelt skulle klikke ind på hver farve i stedet for at evt have en pop menu hvor de kunne ændre farven.
I dette alternativ kunne linke forslag i de forskellige farver bl.a. i bunden af hjemmesiden.

Navn:
Scenarie 5

Identifier:
US03

Forudsætninger:
Forbindelse til databasen ved login
Forbindelse til support

Grundkursus:
User story begynder når kunden skal kontakte vores support for evt spørgsmål.
Kunden skal ind og finde kontakt info via hjemmesiden.
Derefter kan kunden skrive en forspørgelse som bliver sendt til deres support afdeling som en mail.
Support afdelingen modtager mailen og når de besvare mailen bliver den sendt retur til kunden.
User story slutter når kunden har modtaget sit svar fra support.

Alternativ Kursus A:
Alternativ kunne være en live chat så kunden kan skrive live sammen med en medarbejder som sidder klar til at hjælpe .

3. Brugerens krav til database

User stories beskriver kun få af mulige krav til funktioner som alle brugere kunne udnytte. Vi synes at databasen til sådan en e-shop skal opfylde så mange som muligt af kriterierne for både kunder (hjemmesidens brugere) og sælger/admin (Apple). Figur 1 viser flere muligheder som vi vil have at vores database giver til hjemmesidens brugere.

Kunder skal være i stand til:	Sælger skal være i stand til:
1. Oprette en konto, se sine konto data og redigere/slette sine oplysninger	1. Giv mulighed til kunderne at oprette konto og administrere ændringer i dem
2. Se oversigten over alle produkter i kataloget, eller ved kategorier: - navn (fx Mac Book Air) - type (Mac, iMac, iPad) - opløsning (11", 12", 13", etc.) - farve (space grey, sølv, guld) - lagerplads (256 GB, etc.) - pris (i kr.) - in stock	2. Se brugers oversight ved: - navn - username - zip - Dob - bestilte produkter - kontakt oplysninger
3. Vælge produkt(er) til deres indkøbs kurv (shopping "bag")	3. Se oversigten over alle produkter, eller ved kategorier: - navn (fx Mac Book Air) - type (Mac, iMac, iPad) - opløsning (11", 12", 13", etc.) - farve (space grey, sølv, guld) - lagerplads (256 GB, etc.) - pris (i kr.) - in stock

Kunder skal være i stand til:	Sælger skal være i stand til:
4. Oprette en ordre og se sine ordre oplysninger	4. Se alle kunders ordrer ved: - produkt type (Mac Book) - produkt navn - ordre ID - mængde af bestilt produkter - start dato/ afleveringsdato - odres tatus
5. Redigere sin ordre	6. Redigere/slette ordrer
7. Returnere en produkt/ produkter i en ordre	8. Administrere retur på ordrer (se oversigt over returneret ordre)
9. Afbestille produkter	8. Administrere afbestilte ordrer
10. Spore sin ordres afleveringsstatus og history over andre ordrer	9. Svare kundernes spørgsmål omkring ordrer

Bruger-relateret
Produkt-relateret
Ordre-relateret
Admin-support

4. CRUD Matrix

4.1 CRUD beskrivelse

CRUD (Create/Read/Update/Delete) er en tabel, hvor man kan læse de forskellige funktioners rettigheder i ens system. [2] Det er en teknik der ofte bliver brugt til at få et overblik over brugernes adgang til de forskellige opgaver. CRUD kan også bruges fremadrettet til at designe user interface. Alle software programmer bruger CRUD funktionalitet. [3]
Uden CRUD vil et system ikke kunne virke, og CRUD matrix hjælper med at holde styr på dette.

CRUD-matrix nedenunder giver et overblik over forskellige brugers rettigheder i forhold til adskillige klynger af oplysninger: såsom produkter, ordre, brugere og support beskeder. [4]

CRUD MATRIX	Bruger	Sælger	Admin
Log ind/Log ud			
Søg	R	R	R
Produkter	R	R	CRUD
Brugere	CRU	R	CRUD
Support Beskedder	CRU	CRU	CRUD
Ordre	CRU	CRU	CRUD

OBS! Brugeren har kun adgang til sin egne handlinger, hvor en sælger og admin kan gøre noget ved alle handlinger.

5. Database opbygning

For at bygge en database som opfylder alle de bruger-definerede krav vi nævner i Figur 1, skal vi først opdele alle oplysninger omkring kunder og produkter i kategorier, eller tabeller med beskrivelser af forskellige egenskaber, eller attributter. Så organiserer man de tabeller i en data model, som hedder et ER-diagram, på en måde som hjælper til at øge samhørighed mellem data i tabellerne. Denne proces hedder normalization af data og bruges til at ”reducere og helt fjerne dataredundans, fordi det er utroligt svært at gemme objekter i en relationel database, der har de samme oplysninger flere steder.”[5]

Der kan være flere normalization eller normal forms (NF): 1. NF, 2. NF og 3. NF, som viser data optimerings niveau. Jo større er et nummer foran NF, jo mere optimeret er det tilsvarende data schema. Oplysnings kategorierne grupperes i tabeller, som bruges til at lave et ER-diagram tilsvarende til en af de ovennævnte normalization forms. Det er altid bedst at normalisere ER-diagrammet sådan at den svarer til 3. NF og det skal vi gøre i dette projekt.

5. Database opbygning

5.1 ER-diagram - definition og beskrivelse

”ER-diagram, eller entity-relationship data model, er en høj-niveau konceptuel model som beskriver data som enheder, attributter og relationer”.[6] Når man laver et ER-diagram er det for at planlægge og optimere sin database struktur bedst muligt. Oplysningerne opdeles i forskellige tabeller, hvor man definerer hvilken datatype man bruger til de forskellige rækker. Der findes flere forskellige datatyper, som fx INT (Integers, som bruges til tal), VARCHAR (bruges til tekst) og ENUM (bruges i situationer, hvor man får præsenteret forskellige valgmuligheder). Dette er kun nogle få eksempler ud af mange forskellige datatyper.

I ER-diagrammet kigger man på hvilke relationer der er imellem de forskellige tabeller, og hvilken slags relation det er. Der findes 2 forskellige relationer. Identifying relation og Non-Identifying relation. Identifying relation er, hvis en række i en child tabel afhænger af en række i parent tabellen. Dvs. at hvis child tabellen indeholder data fra parent tabellen, kan rækken i child tabellen ikke eksistere, hvis der ikke er data i parent tabellen. Non-Identifying relation er hvor den Primary key fra parent tabellen ikke er nødvendig for at rækken i child tabellen eksisterer. Det vil sige at det er tilladt at have NULL som værdi i rækken.[7]

Udover de 2 forskellige relationer kigger man på om det er en 1 til 1 (1:1), 1 til mange (1:n) eller mange til mange (n:m) relation. Forskellen på disse er at 1 til 1 relation bruges, når man har værdier i 2 tabeller, og man kan sige at der kan være 1 i den ene tabel, og ligeledes kun være 1 i den anden tabel. F.eks. kan man lave 2 tabeller. 1 der indeholder familier, og 1 der indeholder adresser. Her kan man sige at der kan være 1 familie på 1 adresse, og man kan kun have 1 adresse med 1 familie. [8] 1 til mange relation bruges, når man har værdier i 2 tabeller, hvor der kun kan være 1 i den ene, men i den anden kan der være mange. Eksempelvis man kan kun have 1 kunde på en ordre, men 1 kunde kan have mange ordrer. [8]

Mange til mange relation bruges, når man har værdier i 2 tabeller, hvor der kan være flere i begge tabeller. F.eks. kan man have 1 ordre med mange produkter i. Man kan samtidig have 1 produkt i mange ordrer. [8]

Desuden bruger man det der hedder Primary keys og Foreign keys i tabellerne og til relationerne. Primary key repræsenterer den mest vitale række i tabellen og bruges til unikt at identificere hver record i en database tabel. En primary key skal indeholde unikke værdier. Dvs. der ingen duplikerede værdier må være i rækken. Desuden kan en Primary key ikke indeholde NULL værdier. Der kan maximalt være 1 Primary key i en tabel, og de fleste tabeller bør have en Primary key. [9]

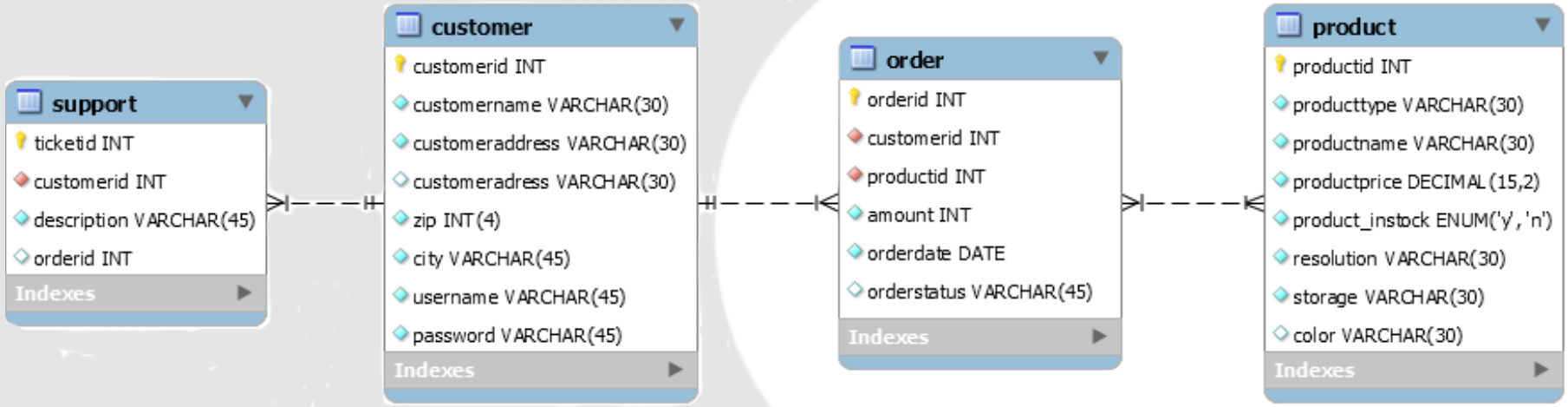
Foreign keys bruges til at lave kryds referencer imellem relateret data i forskellige tabeller og desuden at forhindre handlinger, som kan ødelægge links imellem disse forskellige tabeller. Desuden forhindrer en Foreign key også invalid data i at blive oprettet i en Foreign key række, fordi det skal være en værdi opbevaret i tabellen, som Foreign key peger på.
En Foreign key i en tabel, peger på en Primary key i en anden tabel. [9]

5. Database opbygning

5.2 Udarbejdelse og normalisering af ER-diagram

Tidligere har vi opdelt brugernes krav i 4 kategorier, som kan ses i Figur 1: bruger-relateret (der er 2 bruger typer: kunde og sælger/admin), produkt-relateret, ordre-relateret og admin/support-relateret. Vi har oprettet tilsvarende tabeller: products, customers, orders and support, og har indsat relevante attributer. Tabellernes navne og attribut betegnelser skrev vi på engelsk, da Apple er en international virksomhed og må være i stand til at genbruge databasens struktur. Men selve data i databasen skal være på dansk, da kampagnen er målrettet til danske studerende.

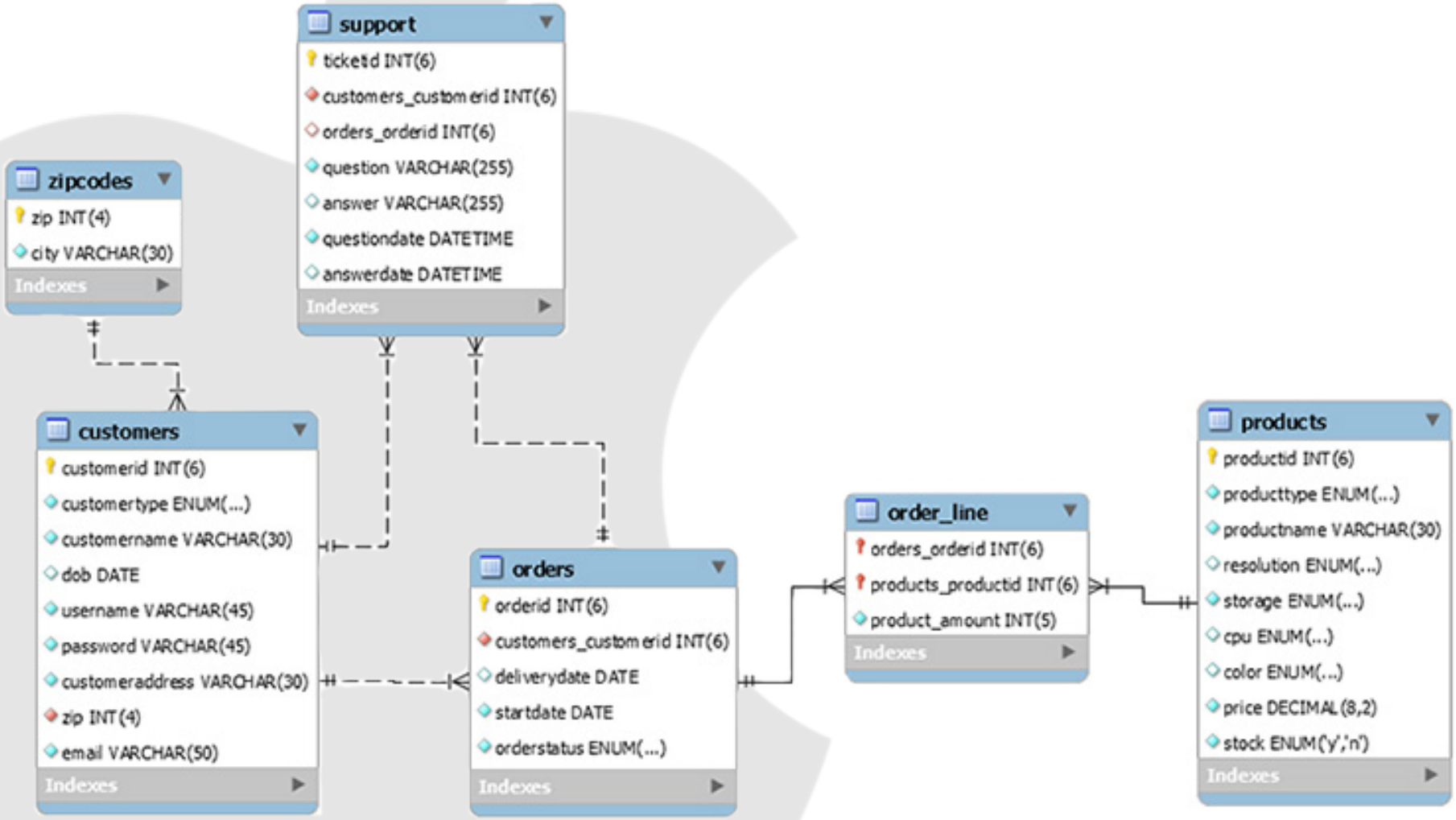
Så lavede vi et simplificeret ER-diagram, som kan ses i Figur 3 nedenunder, for at have en oversigt over alle mulige relationer mellem tabellerne, som vi har oprettet.



5. Database opbygning

Men ER-diagram 1 svarer ikke til 3. NF, da attributterne zip og city i tabellen Customer kan identificeres ved andre attributter end den primære nøgle i den nuværende tabel. Desuden kan man også se, at nogle af dataene kan være der mere end én gang for en specifik primary key. For eksempel, i tabellen Order, kan der være flere produkter i én ordre. Og det kan skabe problemer senere, når vi fx prøver at udregne mængden af produkter for en ordre, som indeholder flere produkter.

Derfor bliver vi nødt til at optimere dette ER-diagram ved at tilsætte flere tabeller, som vist i Figur 4 nedenunder.



5. Database opbygning

Som det kan ses i ER-diagram 2 har vi tilføjet 2 tabeller:

- order_line, som eliminerer mange til mange relation mellem tabellerne orders og products og giver muligheden for at se mængde produkter uanset hvor mange ordre linjer hver ordre har;
- zipcodes, som gør det muligt at undgå gentagne linjer med by navne; hvor zipcoden selv står for primary key.

Således er ER-diagram 2 normaliseret til 3. NF nu. Vi har lavet om på nogle navne i tabeller, fx order til orders, for at undgå problem med lignende Operator linjer i sql koden. Nedenunder beskriver vi relationer mellem tabellerne i diagrammet.

Der er flere relationer mellem Customers og andre tabeller: til zipcodes, til support og til orders. Relationen til zipcodes er lavet for at kunden kun skal indtaste sit postnummer, og at databasen så efterfølgende selv sender dataen ud, til hvilken by kunden bor i. Man kan desuden se at der findes en række der hedder zip i begge tabeller, og i tabellen customers, er zip en foreign key, som binder disse tabeller sammen.

Relationen til support består i customer id, som er primær key i customers, og denne bruges i support tabellen til at identificere brugeren. Dette gør at når der bliver lavet en ticket til support, kan de se hvem brugeren er, og hvilke ordrer og lignende denne bruger har haft. Customerid bliver lavet til en foreign key i support, som hedder customers_customerid.

Relationen til orders er igen customerid, og fungerer på samme måde, som i relationen til support. Desuden har support en relation til orders. Denne er lavet således at kunden kan indtaste sit orderid, og lave en ticket der omhandler den enkelte ordre. I Orders er orderid primary key, og denne bliver så forbundet til support tabellen, så man kan lave den førnævnte ticket.

Orders har så en relation til order_line, som egentligt fungerer som bindeled mellem orders og products databasen. Her er det igen orderid, som bliver brugt som foreign key i order_line, og dette gør at man ved at søge på orderid, kan finde ud af hvilket produkt fra products tabellen der er blevet bestilt, og hvor mange styk af produkterne der er bestilt. Derfor er det også naturligt at productid er bindeleddet imellem order_line og products tabellen, da det er dette man især er interesseret i at få frem i søgninger, hvor man både bruger orderid og productid. Her er begge orderid og productid foreign keys, sådan at tabellen order_line afslutter mange til mange forbindelse mellem products og orders.

ER-diagrammet kan bruges som en database struktur, hvor man følgende kan indsætte data og så sætte databasen i brug.

5. Database opbygning

5.3 Attribut tabel

I attribut tabel kan man vise de samme oplysninger som i et ER-diagram, men uden relationer. Man bruger en attribut tabel til at planlægge, hvordan ens database struktur skal være, og hvilke datatyper man skal bruge i opbygningen af de forskellige tabeller. Dette giver et overblik, og kan lette ens arbejde med databasen en del. Man kan desuden bruge en attribut tabel, til at forklare sin databases indhold til andre, uden at skulle bruge en adgang til databasen. Dette bruges især til dokumentering i virksomheder, og er også en let måde at vedligeholde og planlægge opdateringer af databasen på.

Vi har lavet en attribut tabel, som viser de forskellige tabeller, som vi bruger i vores database og indholdet i disse. Vi har i fællesskab snakket om, hvilket indhold der skal være i de forskellige tabeller, og har valgt de datatyper, som passer bedst til meningen med tabellerne og datatyperne i forhold til brugers krav.

Som man kan se i Appendix 4, har vi valgt at have følgende kolonner:

- Entity (som beskriver hvilke tabeller vi har i databasen),
- Attributes (som beskriver rækkerne i de enkelte tabeller),
- Value (beskriver hvilken værdi man kan indtaste i rækkerne),
- Notes (hvor man beskriver hvis der er noget specielt ved denne række, f.eks. kan det være Unique, som ikke accepterer at der findes duplikater i rækken) og
- Datatype (som enten er Numerisk (Tal) eller Alfanumerisk (Karakterer).

5.4 Databasens indhold

Nu når vi har databasens struktur klar i form af ER-diagram 2, kan vi sætte data ind ved hjælp af Workbench, som vi har brugt til at lave vores ER-diagrammer.
Vi har lavet en meget forkortet katalog af produkter som vi synes passer bedst til en kampagne til studerende. Vi har også tænkt på at skoler kan være interesseret i at købe Apple computers til skolestart. Og de ville købe flere stykke ad gangen, som er godt for Apple. Derfor indeholder kataloget også stor-skærm computers, samt iPads. Kataloget kan ses i Appendix 5.
Vi har indsat alle produkter (34) fra kataloget til tabellen products. Der er mindre data i andre mapper dog,bare nok for at nå varierede svarer, når vi tjekker vores database senere. Mere udvidet indhold af data kan ses i sql-koden.

6. SQL-kode

Heldigvis er det let at generere sql-koden ud fra ER-diagram lavet i Workbench. Det gør man ved hjælp af Forward Engineer funktion i Menu-feltet Database. Det gjorde vi og fik koden med det samme. Man skal bare huske at markere afkrydsningsfeltet "Generate INSERT statements for tables". Og så skrev vi alle kommentarer til koden. Den færdige sql-kode kan ses i vedhæftet fil.

6.1 Testning af SQL-koden

For at sikre at al koden er skrevet rigtigt og alle data kan vises, skulle vi udføre flere forespørgsler. Det gjorde vi ved hjælp af SELECT statements og forskellige sorting (ORDER BY clauses) og filtering (WHERE clauses) funktioner.

6.2 Views

Vi har organiseret resultater i Views, som er virtuelle tabeller, som indeholder forespørgsler, der dynamisk henter data, når views er aktiveret [3]. Views ligger i en separate mappe, når man åbner koden i Workbench og viser kun forespørgsels resultater.
Først lavede vi en liste af spørgsmål eller opgaver for data hentning til Views, som er vist i Appendix 6. Vi har lavet 2-4 forspørgsler til hver tabel i vores ER-diagram. Views er med i koden selv. Appendix 7 viser nogle eksempler på Views og data hentnings resultater.
Til fleste forespørgsler brugte vi SELECT, FROM, WHERE and ORDER BY clauses [10]:

```
SELECT customerid, customername, orderid, orderstatus
FROM customers, orders
WHERE customers_customerid=customerid AND orderstatus = 'cancelled'
ORDER BY customerid;
```

6.3 Fejl rettelse i sql-koden

Under testning af SQL-koden, opstod der flere fejl med databasen og vi skulle retter på ER-diagrammet flere gange. For eksempel, hvis man glemmer at skrive join condition (WHERE…) eller indtaster foreign keys forkert i den, så får man et problem som hedder Cartesian Product. Problemet er nemlig at man får for mange linjer af de samme oplysninger og nogle gange slet ikke relevante til forespørgslen. Det sker, fordi antallet af rækker bliver multipliceret på tværs af forskellige tabeller. I sådan tilfælde kan resultaterne ikke bruges til noget. [11]
Der opstår også et problem med syntax, når en tabels eller datatypes navn er sammenfaldende med nogle andre statements eller clauses i sql. For eksempel, kunne vi ikke bruge order som navn på tabellen med ordres, fordi der findes allerede en clause ORDER BY i sql og ordet bliver læst som en kammand i stedet for data entry. Så ændrede vi tabellens navn til orders.

7. Konklusion

Igennem projektets forløb har vi lært at bruge mysql Workbench, kode mySQL, teste via Views, forstå og opbygge relationer mellem forskellige typer data,blandt andet med hjælp af ER-diagrams, samt at lave CRUD-matrix og beskrive User stories. Vi er allesammen blevet klogere på hvordan en mysql database bliver opbygget og designet, og alle de processer man skal igennem.
Forløbet er gået godt, vi har formået at samarbejde og møde samlet op til vores møder samt arbejde og fordelt opgaver på skolen. Vi har opnået vores mål, som var at skabe en velfungerende database tilsvarende bruger-definiret krav relevant til en udvalgt e-shop. Desuden nåede vi at inkluderer ekstra arbejdsmateriale til vores rapport, såsom PB og WBS, Attribut tabel, Views, osv.

Vi havde nogle udfordringer under projektets forløb. For eksempel var flere møder aflyst pga. sygdom, så kunne vi ikke altid holde til planlagt frister. Vi var nødt til at lave om på rækkefølgen på nogle af opgaverne, som kan ses på rapporten, hvor vi præsentere forskellige modeller i en anden rækkefølge i forhold til planen (Gantt-chart).
Derudover har har vi haft udfordringer med at samle rapporten da der er udstået noget fejl kommunikation samt deadlines som er blevet overskredet, som nok kunne have udbedres ved mere grundig gennemgang af strukturering samt bedring i forhold til at stille spørgsmål ved evt. tvivl.

Vi ville rigtig gerne have brugt vores database i en rigtig sammenhæng på en hjemmeside, og det ville også være en god mulighed som et ydeligere projekt.

8. Referenceliste

[1] http://www.apple.com/dk/shop/browse/campaigns/education_pricing

[2] http://infocenter.sybase.com/archive/index.jsp?topic=%2Fcom.sybase.stf.powerdesigner.docs_12.0.0%2Fhtml%2Fbpug%2Fbpugp124.htm

[3] https://nl.wikipedia.org/wiki/CRUD

[4] http://blog.aplikacja.info/2011/12/crud-matrix-as-a-software-design-and-estimation-tool/

[5] http://agiledata.org/essays/dataNormalization.html “Introduction to Data Normalization: A Database “Best” Practice” af Scott Wybler, 2013

[6] ”Data Modeling with Entity-Relationship diagrams” af Riccardi, 06-28-2002, s. 62

[7] https://dev.mysql.com/doc/workbench/en/wb-relationship-tools.html

[8] http://code.tutsplus.com/articles/sql-for-beginners-part-3-database-relationships--net-8561

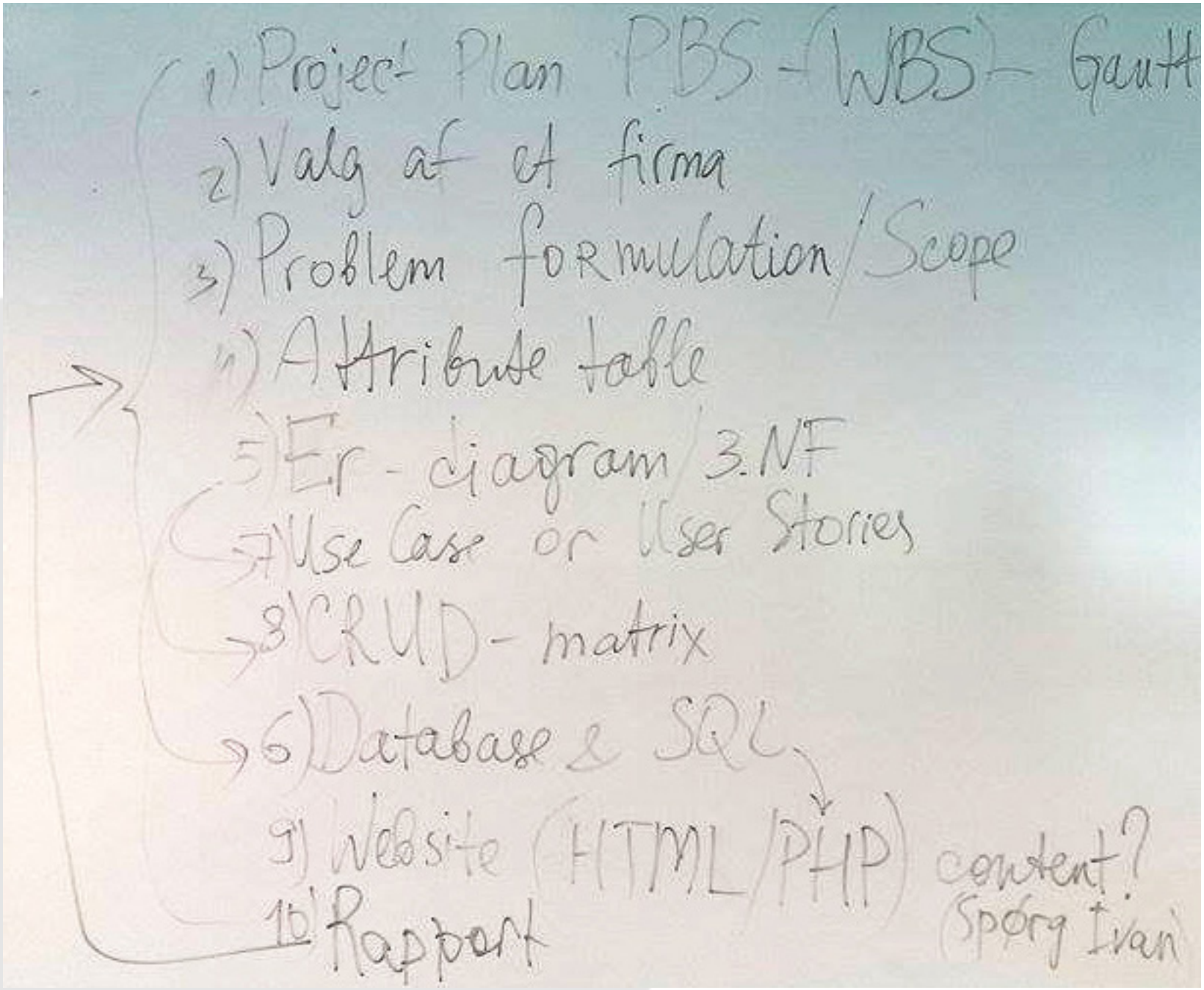
[9] http://www.w3schools.com/sql/sql_constraints.asp

[10] “SQL in 10 minutes” af Ben Forta, 2004, ss. 13-15, 22, 27-29, 33-34

[11] “SQL in 10 minutes” af Ben Forta, 2004, ss. 95-97

9. Appendix

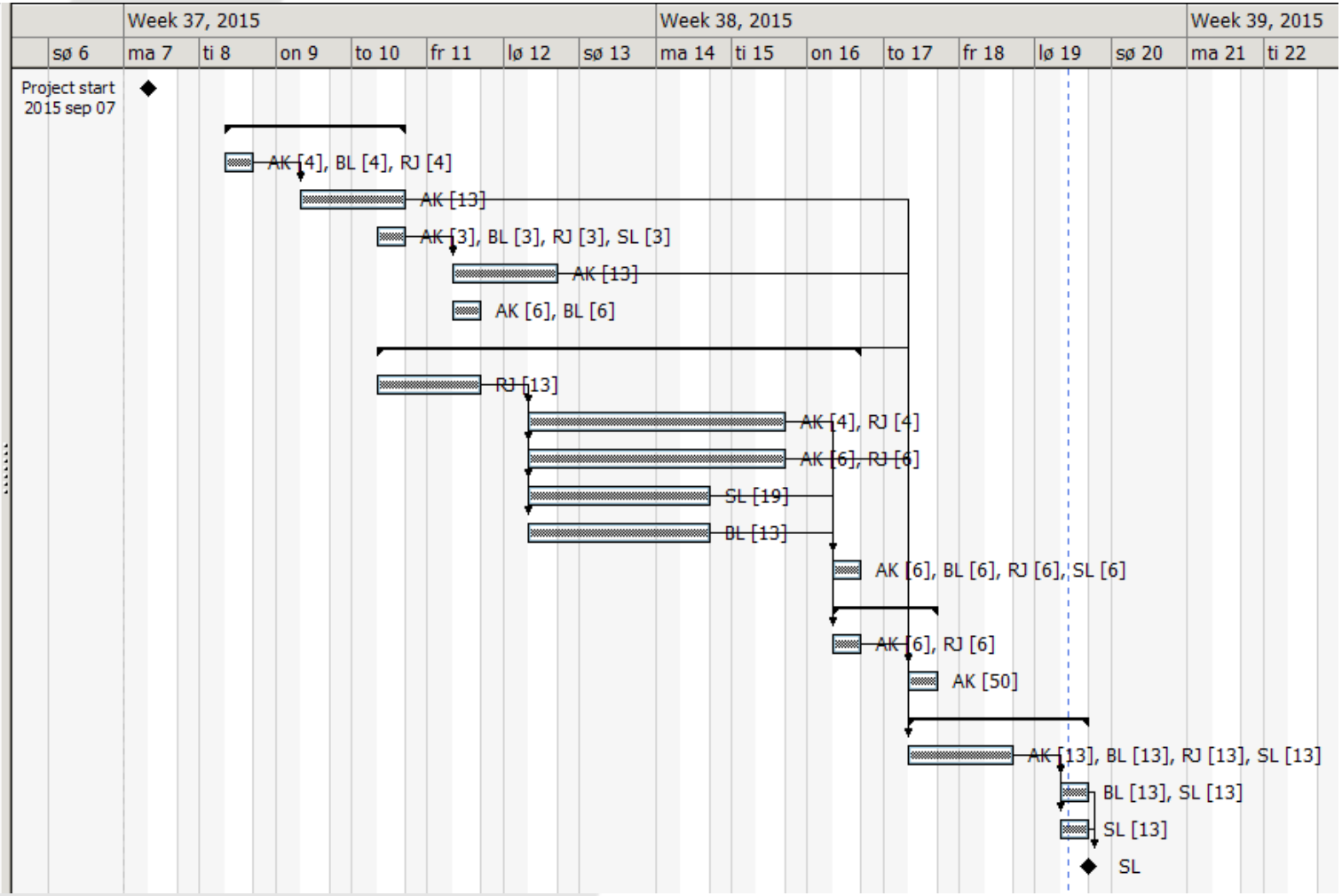
Appendix 1



9. Appendix

Appendix 2

WBS	Name	Work
1	Opgave aflevering	N/A
2	**Project planlægni**	**3h**
2.1	PBS	1h
2.2	Gantt-chart	2h
3	Firma valg	1h
4	Problem formulerin	2h
5	Brugers krav	1h
6	**Modeller og datal**	**1d 6h**
6.1	Attribute table	2h
6.2	Database	2h
6.3	ER-diagram/3.NF	3h
6.4	User stories	3h
6.5	CRUD-matrix	2h
6.6	Tjek alle modelle	2h
7	**Kodning**	**5h**
7.1	SQL i Workbench	1h
7.2	SQL testning	4h
8	**Rapport**	**1d 2h**
8.1	Rapport skrivning	1d
8.2	Rapport redigeri	2h
9	Foreberedelse af fil	1h
10	Aflevering	N/A



9. Appendix

Appendix 3

WBS	Name	Start	Finish	Work	Duration	Slack	Cost	Assigned to	% Complete
1	Opgave aflevering	sep 7	sep 7	N/A	N/A	12d	0		100
2	**Project planlægning**	**sep 8**	**sep 10**	**3h**	**3d**	**8d**	**0**		100
2.1	PBS	sep 8	sep 8	1h	1d	5d	0	AK, BL, RJ	100
2.2	Gantt-chart	sep 9	sep 10	2h	2d	5d	0	AK	100
3	Firma valg	sep 10	sep 10	1h	1d	3d	0	AK, BL, RJ, SL	100
4	Problem formulering	sep 11	sep 12	2h	2d	3d	0	AK	100
5	Brugers krav	sep 11	sep 11	1h	1d	7d	0	AK, BL	100
6	**Modeller og database**	**sep 10**	**sep 16**	**1d 6h**	**6d**		**0**		100
6.1	Attribute table	sep 10	sep 11	2h	2d		0	RJ	100
6.2	Database	sep 12	sep 15	2h	3d		0	AK, RJ	100
6.3	ER-diagram/3.NF	sep 12	sep 15	3h	3d		0	AK, RJ	100
6.4	User stories	sep 12	sep 14	3h	2d	1d	0	SL	100
6.5	CRUD-matrix	sep 12	sep 14	2h	2d	1d	0	BL	100
6.6	Tjek alle modeller	sep 16	sep 16	2h	1d		0	AK, BL, RJ, SL	100
7	**Kodning**	**sep 16**	**sep 17**	**5h**	**2d**	**2d**	**0**		100
7.1	SQL i Workbench	sep 16	sep 16	1h	1d	2d	0	AK, RJ	100
7.2	SQL testning	sep 17	sep 17	4h	1d	2d	0	AK	100
8	**Rapport**	**sep 17**	**sep 19**	**1d 2h**	**3d**		**0**		100
8.1	Rapport skrivning	sep 17	sep 18	1d	2d		0	AK, BL, RJ, SL	100
8.2	Rapport redigering	sep 19	sep 19	2h	1d		0	BL, SL	100
9	Foreberedelse af filer	sep 19	sep 19	1h	1d		0	SL	100
10	Aflevering	sep 19	sep 19	N/A	N/A		0	SL	100

9. Appendix

Appendix 4

Entity	Attributes	Value	Notes	Datatype (N/AN)
Customers	customerid	1-X	Max 6 numbers & Auto Increment	N
	customertype	Student/School	Either student or school	A/N
	customername	a-Å	Max 30 characters	A/N
	dob	date	birth day of user	N
	username	All Characters	Unique Max 45 characters	A/N
	password	All Characters	Max 45 characters	A/N
	customeraddress	All Characters	Max 30 characters	A/N
	zip	1-X	Max 4 numbers	N
	email	All Characters	Unique & Max 50 characters	A/N
Products	productid	1-X	Max 6 numbers & Auto Increment	N
	producttype	Bærbar/Storskærm/iPad	Either bærbar, storskærm or iPad	A/N
	productname	a-Å	Max 30 characters	A/N
	resolution	11"/12"/13"/21,5"/27"	Either 11", 12", 13", 21,5" or 27"	N
	storage	16 GB/32 GB/64 GB/128 GB/256 GB/500 GB/512 GB/1 TB	Either 16 GB, 32 GB, 64 GB, 128 GB, 256 GB, 500 GB, 512 GB or 1 TB	N
	storage	17 GB/32 GB/64 GB/128 GB/256 GB/500 GB/512 GB/1 TB	Either 16 GB, 32 GB, 64 GB, 128 GB, 256 GB, 500 GB, 512 GB or 1 TB	N
	storage	18 GB/32 GB/64 GB/128 GB/256 GB/500 GB/512 GB/1 TB	Either 16 GB, 32 GB, 64 GB, 128 GB, 256 GB, 500 GB, 512 GB or 1 TB	N
	price	1-X	Max 8 numbers before comma & Max 2 numbers after comma	N
	stock	Y/N	Either y or n	A/N
Orders	orderid	1-X	Max 6 numbers & Auto Increment	N
	customers_customerid	1-X	Max 6 numbers	N
	products_productid	1-X	Max 6 numbers	N
	deliverydate	1-X	DATE, when delivered (anbefalet TIMESTAMP Current date)	N
	startdate	1-X	DATE, when ordered (anbefalet TIMESTAMP Current date)	N
	orderstatus	Incart/Purchased/Cancelled/Returned	incart, inprocess, purchased, cancelled or returned	A/N
Order_line	orders_orderid	1-X	Max 6 numbers	N
	products_productid	1-X	Max 6 numbers	N
	product_amount	1-X	Max 5 numbers	N
Support	ticketid	1-X	Max 6 numbers & Auto Increment	N
	customers_customerid	1-X	Max 6 numbers	N
	orderid	1-X	Max 6 numbers	N
	question	a-Å	Max 255 characters	A/N
	answer	a-Å	Max 255 characters	A/N
	questiondate	date	DATETIME when question is created (anbefalet TIMESTAMP Current date)	N
	answerdate	date	DATETIME when question is answered (anbefalet TIMESTAMP Current date)	N
Zipcodes	zip	1-X	Unique & Max 4 numbers	N
	city	a-Å	Max 30 characters	A/N

9. Appendix

Appendix 5

APPLE Produkt katalog

Produkt id	Produktnavn	Produkttype	Skærm opløsning (resolution)	Lagerplads (storage capacity)	CPU capacitet	Farve (sølv/ space grey/ guld)	Pris (kr.)	På lager (yes/no)
1	Mac Book	bærbar	12"	256	1,1 GHz	sølv	10621,25	y
2	Mac Book	bærbar	12"	512	1,2 GHz	sølv	13158,75	y
3	Mac Book	bærbar	12"	256	1,1 GHz	space grey	10621,25	y
4	Mac Book	bærbar	12"	512	1,2 GHz	space grey	13158,75	y
5	Mac Book	bærbar	12"	256	1,1 GHz	guld	10621,25	n
6	Mac Book	bærbar	12"	512	1,2 GHz	guld	13158,75	n
7	Mac Book Air	bærbar	11"	128	1,6 GHz	sølv	7331,25	y
8	Mac Book Air	bærbar	11"	256	1,6 GHz	sølv	8,928,75	y
9	Mac Book Air	bærbar	13"	128	1,6 GHz	sølv	8271,75	y
10	Mac Book Air	bærbar	13"	256	1,6 GHz	sølv	9868,25	y
11	Mac Book Pro	bærbar	13"	500	2,5 GHz	sølv	8928,75	y
12	iMac	storskærm	21,5"	500	1,4 GHz		8928,75	y
13	iMac	storskærm	21,5"	1000	2,7 GHz		10621,25	y
14	iMac	storskærm	27"	1000	3,2 GHz		15038,75	n
15	iPad Air	ipad		16		sølv	3838,75	n
16	iPad Air	ipad		32		sølv	4222,5	y
17	iPad Air	ipad		16		space grey	3838,75	y
18	iPad Air	ipad		32		space grey	4222,5	y
19	iPad Air 2	ipad		64		sølv	5375	y
20	iPad Air 2	ipad		128		sølv	6142,5	y
21	iPad Air 2	ipad		64		space grey	5375	n
22	iPad Air 2	ipad		128		space grey	6142,5	n
23	iPad Air 2	ipad		64		guld	5375	y
24	iPad Air 2	ipad		128		guld	6142,5	y
25	iPad mini 2	ipad		16		sølv	3167,5	y
26	iPad mini 2	ipad		32		sølv	3455	y
27	iPad mini 2	ipad		16		space grey	3167,5	y
28	iPad mini 2	ipad		32		space grey	3455	y
29	iPad mini 4	ipad		64		sølv	4607,5	y
30	iPad mini 4	ipad		128		sølv	5375	n
31	iPad mini 4	ipad		64		space grey	4607,5	y
32	iPad mini 4	ipad		128		space grey	5375	y
33	iPad mini 4	ipad		64		guld	4607,5	n
34	iPad mini 4	ipad		128		guld	5375	y

9. Appendix

Appendix 6

Views forspørgsler:

1. Customers:

Find alle kunder som købte iPads sidste måned.
Hvor mange kunder købte Mac Book i denne måned?
Find kunder som har aflyst deres ordrer? Hvorfor?
Hvilken type kunder købte fleste produkter?

2. Products:

Find alle produkter som er ikke afleveret endnu i denne måned?
Find alle produkter der var afbestilt sidste måned.
Find alle produkter som er billigere end 10,000 kr.
Find alle produkter som har storage på 128 GB.

3. Orders:

Hvilke produkter har CPHbusiness-Lyngby bestilt? Var nogle af dem afbestilt (cancelled)? Hvilke?
Hvilke produkter har CPHbusiness-City bestilt? Var nogle af dem afbestilt (cancelled)?

4. Support:

Find alle kunder, der har sendt spørgsmål til support og hvornår.
Hvilken kunde kan ikke se sin ordre? Hvorfor?
Var alle spørgsmål svaret den samme dag de var posted? Hvis ikke, hvilket spørgsmål var ikke svaret den samme dag?

5. Zip:

Find alle ordrer der var lavet på 2800.
Find alle kunder som købte sølv iPad Air 128 GB? Hvilken by er de fra?

9. Appendix

Appendix 7

Eksempler på tabeller i sql-koden: (Appendix 7_Koden 1)

```
-- Creates a table `customers` in `mydb`. First drops a previous table with such name if exists.
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`customers` ;

CREATE TABLE IF NOT EXISTS `mydb`.`customers` (
  `customerid` INT(6) NOT NULL AUTO_INCREMENT COMMENT '',
  `customertype` ENUM('student', 'school') NOT NULL COMMENT '',
  `customername` VARCHAR(30) NOT NULL COMMENT '',
  `dob` DATE NULL COMMENT '',
  `username` VARCHAR(45) NOT NULL COMMENT '',
  `password` VARCHAR(45) NOT NULL COMMENT '',
  `customeraddress` VARCHAR(30) NOT NULL COMMENT '',
  `zip` INT(4) NOT NULL COMMENT '',
  `email` VARCHAR(50) NOT NULL COMMENT '',
  PRIMARY KEY (`customerid`)  COMMENT '',
  CONSTRAINT `fk_customers_zip_zipcodes1`
    FOREIGN KEY (`zip`)
    REFERENCES `mydb`.`zipcodes` (`zip`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

Eksempler på indtastede data i sql-koden: (Appendix 7_Koden 2)

```
-- Inserts data into table `mydb`.`zipcodes`
-- -----------------------------------------------------
START TRANSACTION;
USE `mydb`;
INSERT INTO `mydb`.`zipcodes` (`zip`, `city`) VALUES (1119, 'Landemærket');
INSERT INTO `mydb`.`zipcodes` (`zip`, `city`) VALUES (2300, 'København S');
INSERT INTO `mydb`.`zipcodes` (`zip`, `city`) VALUES (2700, 'Brønshøj');
INSERT INTO `mydb`.`zipcodes` (`zip`, `city`) VALUES (2800, 'Kongens Lyngby');
INSERT INTO `mydb`.`zipcodes` (`zip`, `city`) VALUES (3000, 'Helsingør');
INSERT INTO `mydb`.`zipcodes` (`zip`, `city`) VALUES (3200, 'Helsinge');

COMMIT;
```

9. Appendix

Appendix 7

Eksempler på views og hentede resultater:

Eksempel 1: customers-view 1 (Appendix 7_View 1)

```
-- Opret view som viser alle kunder (id, navn og købsdato) som købte producttype 'ipad' sidst
-- Først sletter vi en view med det samme navn, hvis den eksisterer.

DROP VIEW IF EXISTS `mydb`.`customersview1`;

CREATE VIEW customersview1 AS
SELECT customerid, customername, startdate
FROM customers, products, orders, order_line
WHERE customers_customerid=customerid AND products_productid=productid AND orders_orderid=ord
AND producttype = 'ipad' AND startdate BETWEEN '2015-07-31' AND '2015-09-01'
ORDER BY customerid;
```

Resultat på customers-view 1 (Appendix 7_View 1-result)

customerid	customername	startdate
2	Rene Jensen	2015-08-25
3	Sabine Larsen	2015-08-26
5	Benjamin Lipsø	2015-08-27

Eksempel 2: Products-view 4 (Appendix 7_View 2)

```
-- Opret view som viser alle produkter som har storage på 128 GB.
-- Først sletter vi en view med det samme navn, hvis den eksisterer.

DROP VIEW IF EXISTS `mydb`.`productsview4`;

CREATE VIEW productsview4 AS
SELECT productid, producttype, productname, resolution, `storage`
FROM products
WHERE `storage` = '128 GB'
ORDER BY productid;
```

Resultat på products-view 4 (Appendix 7_View 2-result)

9. Appendix

Appendix 7.1 & 7.2

```
-- Inserts data into table `mydb`.`zipcodes`
-- -----------------------------------------------------
START TRANSACTION;
USE `mydb`;
INSERT INTO `mydb`.`zipcodes` (`zip`, `city`) VALUES (1119, 'Landemærket');
INSERT INTO `mydb`.`zipcodes` (`zip`, `city`) VALUES (2300, 'København S');
INSERT INTO `mydb`.`zipcodes` (`zip`, `city`) VALUES (2700, 'Brønshøj');
INSERT INTO `mydb`.`zipcodes` (`zip`, `city`) VALUES (2800, 'Kongens Lyngby');
INSERT INTO `mydb`.`zipcodes` (`zip`, `city`) VALUES (3000, 'Helsingør');
INSERT INTO `mydb`.`zipcodes` (`zip`, `city`) VALUES (3200, 'Helsinge');

COMMIT;
```

customerid	customername	customertype	orderid	sum(product_amount)
4	CPHbuisness-Lyngby	school	2	25
4	CPHbuisness-Lyngby	school	3	25
6	CPHbusiness-City	school	5	46
6	CPHbusiness-City	school	6	48

9. Appendix

Appendix 7.3 & 7.4

```
-- Opret view som viser hvilken type kunder (id, type) købte flere end 20 produkter per order.
-- Først sletter vi en view med det samme navn, hvis den eksisterer.

DROP VIEW IF EXISTS `mydb`.`customersview4`;

CREATE VIEW customersview4 AS
SELECT customerid, customername, customertype, orderid, sum(product_amount)
FROM customers, orders, order_line
WHERE customers_customerid=customerid AND orders_orderid=orderid
GROUP BY orderid
HAVING sum(product_amount) > 20;
```

productid	producttype	productname	orderid	orderstatus
4	bærbar	Mac Book	2	inprocess
3	bærbar	Mac Book	2	inprocess
6	bærbar	Mac Book	3	cancelled
5	bærbar	Mac Book	3	cancelled

9. Appendix

Appendix 7.5 & 7.6

```
-- Opret view som viser alle kunder, der har sendt spørgsmål til support og hvornår.
-- Først sletter vi en view med det samme navn, hvis den eksisterer.

DROP VIEW IF EXISTS `mydb`.`supportview1`;

CREATE VIEW supportview1 AS
SELECT ticketid, customerid, customertype, customername, question, questiondate
FROM customers, support
WHERE customerid = customers_customerid
ORDER BY ticketid;
```

orderid	customername	productname	storage	color	zip	city
6	CPHbusiness-City	iPad Air 2	storage	sølv	1119	Landemærket
7	Rene Jensen	iPad Air 2	storage	sølv	3200	Helsinge

9. Appendix

Appendix 7.7 & 7.8

```
-- Opret view som viser alle kunder som købte sølv iPad Air 2 med storage på 128 GB.
-- Hvilke by(er) er de fra? Først sletter vi en view med det samme navn, hvis den eksisterer.

DROP VIEW IF EXISTS `mydb`.`zipcodesview2`;

CREATE VIEW zipcodesview2 AS
SELECT orderid, customername, productname, 'storage', color, zipcodes.zip, city
FROM zipcodes, orders, order_line, customers, products
WHERE customerid = customers_customerid AND orderid = orders_orderid AND productid = products_productid
AND zipcodes.zip = customers.zip AND productname='iPad Air 2' AND `storage`='128 GB' AND color='sølv'
ORDER BY orderid;
```

customerid	customername	startdate
2	Rene Jensen	2015-08-25
3	Sabine Larsen	2015-08-26
5	Benjamin Lipsø	2015-08-27

9. Appendix

Appendix 7.9 & 7.10

```sql
-- Opret view som viser alle kunder (id, navn og købsdato) som købte producttype 'ipad' sidste måned.
-- Først sletter vi en view med det samme navn, hvis den eksisterer.

DROP VIEW IF EXISTS `mydb`.`customersview1`;

CREATE VIEW customersview1 AS
SELECT customerid, customername, startdate
FROM customers, products, orders, order_line
WHERE customers_customerid=customerid AND products_productid=productid AND orders_orderid=orderid
AND producttype = 'ipad' AND startdate BETWEEN '2015-07-31' AND '2015-09-01'
ORDER BY customerid;
```

```sql
-- Opret view som viser alle produkter som har storage på 128 GB.
-- Først sletter vi en view med det samme navn, hvis den eksisterer.

DROP VIEW IF EXISTS `mydb`.`productsview4`;

CREATE VIEW productsview4 AS
SELECT productid, producttype, productname, resolution, `storage`
FROM products
WHERE `storage` = '128 GB'
ORDER BY productid;
```

9. Appendix

Appendix 7.11

```
-- Creates a table `customers` in `mydb`. First drops a previous table with such name if exists.
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`customers` ;

CREATE TABLE IF NOT EXISTS `mydb`.`customers` (
  `customerid` INT(6) NOT NULL AUTO_INCREMENT COMMENT '',
  `customertype` ENUM('student', 'school') NOT NULL COMMENT '',
  `customername` VARCHAR(30) NOT NULL COMMENT '',
  `dob` DATE NULL COMMENT '',
  `username` VARCHAR(45) NOT NULL COMMENT '',
  `password` VARCHAR(45) NOT NULL COMMENT '',
  `customeraddress` VARCHAR(30) NOT NULL COMMENT '',
  `zip` INT(4) NOT NULL COMMENT '',
  `email` VARCHAR(50) NOT NULL COMMENT '',
  PRIMARY KEY (`customerid`)  COMMENT '',
  CONSTRAINT `fk_customers_zip_zipcodes1`
    FOREIGN KEY (`zip`)
    REFERENCES `mydb`.`zipcodes` (`zip`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

9. Appendix

Appendix 8

Study: Multimediea Design
Project 3.semester 1. Project: Database
Period: 07. Sep. 2015 – 20. Sep. 2015

Fact Sheet

Project title:	Projekt: Database	
Class:	MulA & MulB	
Group Number:	MulB07	
Group member:		
Name:	School mail:	Sign:
Anastasia Keller (MulA)	cph-ak186@cphbusiness.dk	
René Abel Jensen (MulB)	cph-rj135@cphbusiness.dk	
Benjamin Lipsø (MulB)	cph-bl91@cphbusiness.dk	
Sabine Larsen (MulB)	cph-sl176@cphbusiness.dk	

By signing this document we confirm that the submitted material is all our own work. We have not copied another person's work or used material without referencing the source. We have not cheated in any way.