Joachim Grimen Westgaard

Databaser

Øving 8

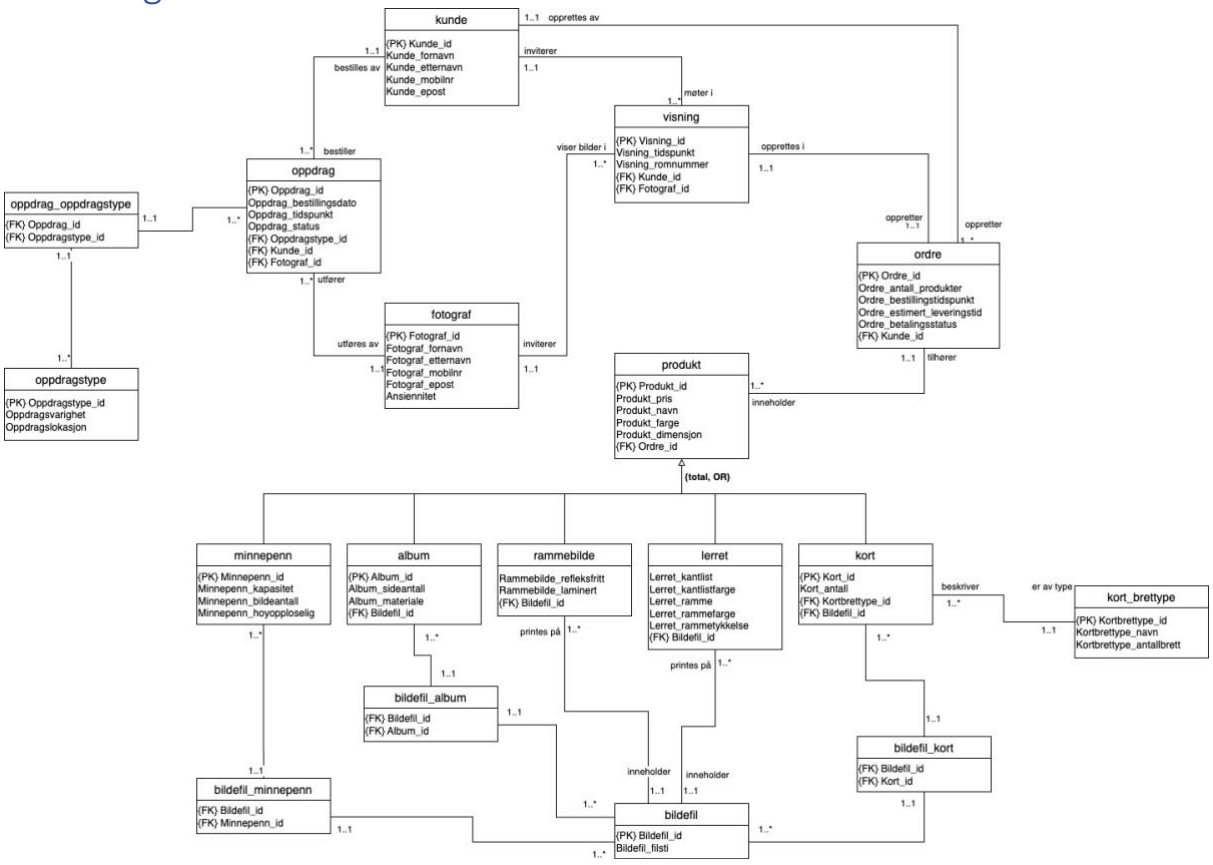
1 Oppgavebeskrivelse

Databasen skal representere en oversiktlig arbeidssituasjon hos TiTT Melhuus AS, en fotografbedrift.
Formålet med databasen er å gi bedriften en heldigital løsning på nåværende papirløsninger angående informasjonslagring.
Bedriften ønsker å kunne referere til databasen for å kunne finne fram oppdrag og planlegge fotografenes uker.

Bedriften ønsker å lagre grunnleggende kundeinformasjon. En kunde bestiller et oppdrag, som utføres av én av fotografene. Det finnes flere ulike typer oppdrag, slik som studio- og utendørsfotograferinger. Etter et oppdrag er gjennomført, skal fotografen klargjøre de beste bildende og vise disse frem til kunden (ved en senere anledning) under en såkalt visning. Her får kunden mulighet til å bestille en ordre av ulike produkter som bedriften selger.
Følgende produkter tilbys: minnepenn (lavoppløselig/høyoppløselig), fotoalbum, rammebilde, lerret og kort.
Alle produktene (foruten om minnepenner) vil ha en dimensjon. Et rammebilde bruker alltid glass, men kan også være refleksfritt. Et lerret kan ha en såkalt kantlist eller ramme. Et kort kan for eksempel være et invitasjonskort eller et takkekort. Det spesifiseres ulike typer bretter på disse kortene. Alle produktene baserer seg på bildefilene tatt av fotografen under oppdraget.

Databasen er relativt statisk oppbygd, i den forstand at den kun har de mest åpenbare relasjonene mellom entitetstypene implementert.

2 ER-diagram



3 Databasen i MySQL

3.1 Eksempler på spørringer

Finne alle oppdrag en fotograf har i løpet av en gitt periode:

```
SELECT * FROM oppdrag
WHERE oppdrag.oppdrag_tidspunkt
BETWEEN '2023-11-20' AND '2023-11-24'
JOIN fotograf ON oppdrag.fotograf_id = fotograf.fotograf_id
WHERE oppdrag.fotograf_id = 1
```

Finne alle ordrene en kunde har bestilt:

```
SELECT * FROM ordre
JOIN kunde ON ordre.kunde_id = kunde.kunde_id
```

Finne alle lerretene som hører til i en spesifikk ordre:

```
SELECT * FROM produkt
WHERE produkt.produkt_navn = 'Lerret'
JOIN ordre ON produkt.ordre_id = ordre.ordre_id
WHERE produkt.ordre_id = 101
```

3.2 Oppsummering

Da jeg ikke fullstendig implementerte databasen, men kun en skissering av den, vil det ikke være mulig å teste spørringene. Dette er noe jeg ønsker å finne tid til senere.

Databasen inkluderer ikke absolutt alle elementene bedriften er nødt til å holde styr på, men de mest essensielle.
Spørringene var relativt enkle å lage da jeg brukte god tid på å modellere ER-diagrammet og entitetene sine attributter og relasjoner. Dette tar jeg med meg videre som god lærdom.

3.3 Videre arbeid

Dersom jeg skulle ha videreutviklet databasen, ville jeg nok ha jobbet tettere opp mot bedriften og vært i dialog med de under modelleringen, slik at alle mulige momenter hos bedriften ble dekket i databasen. I tillegg ville jeg ha realisert databasen i MySQL for testing.

4 XML/JSON

Dersom vi ønsker å lagre en sammensatt verdi i et attributt, kan vi benytte oss av en implementasjon av semistrukturell data. Vi kan for eksempel ønske å fylle inn dimensjonene til et produkt i den skisserte databasen ovenfor. Hvilket mål som er høyde, bredde og tykkelse kan variere etter personen som skrev inn dimensjonene i databasen.

I SQL kan vi benytte oss av følgende kode for å sette inn en kolonne med sammensatt verdi:

```
ALTER TABLE produkt
ADD COLUMN produkt_dimensjon XML;
```

Her spesifiseres det at kolonnen skal kunne ta imot semistrukturell data som input. Dette kan derfor også ses på som en ekstra verifisering i datastrukturen som angis i skjemaet.
Et eksempel på en INSERT-setning kan i tabellen produkt se ut som følger:

```
INSERT INTO produkt (produkt_id, produkt_pris, produkt_navn,
produkt_farge, produkt_dimensjon) VALUES
(1, 123, 'Lerret', 'Gul',
'<produkt><dimensjon>
<hoyde>50</hoyde>
<bredde>70</bredde>
<dybde>5</dybde>
</dimensjon></produkt>');
```

Her består altså utdanningen til Joachim Westgaard av den videregående skolen Lambertseter VGS og universitetet NTNU. Vi kan si at XML forenkler organiseringen av hierarkisk strukturer i databasen vår.

Dersom vi ønsker å lage en liste over alle produkter med en høyde-dimensjon på 50 (cm), så kan vi kjøre følgende spørring:

```
SELECT * FROM produkt
WHERE ExtractValue(produkt_dimensjon, '/produkt/dimensjon/hoyde') =
50;
```

En av fordelene med XML i databasen er som sagt at vi enklere kan organisere hierarkiske strukturer med mye intern variasjon, slik som for eksempel et produkt sine dimensjoner eller kontaktinformasjon til en person (epost(er), mobilnummer(e)...).
En ulempe med dette kan være at databasen mister litt ytelse i store datasett, da XML krever litt mer prosesseringskraft sammenlignet med standard datatyper.
I tillegg så kan JSON være mer lesbar og moderne sammenlignet med XML.

5 NoSQL

Dersom bedriften forventer en stor økning i antall oppdrag eller kunder og dermed en stor økning i databasen, så kan skalerbarheten til en NoSQL-løsning vise seg å være mer effektiv og gi bedre ytelse.
Derfor kunne det ha vært en mulighet å lagre disse tabellene i en NoSQL-database.

En NoSQL-database kan være raskere og mindre kompleks å implementere. I tillegg kan den være mer mottakelig for strukturelle endringer over tid, noe som kan være fornuftig dersom bedriften ser for seg en overhaling av den interne strukturen over tid.

Dersom vi oppdager at databasen ikke krever svært kompliserte relasjoner, så kan en NoSQL-løsning være mer optimalt, da disse typene databaser fungerer godt under disse omstendighetene.