DB - øving 8

Vi jobbet sammen, men leverer samme filer begge to
Ingebrigt Kristoffer Thomassen Hovind og Erling Sung Sletta

1.1.Definere behov

Beskriv kort (maks. en side) eget case/egen problemstilling og klargjør målene for lagring og representasjon av data knyttet til case-en. Hva ønsker du å oppnå? Finnes det begrensninger i hva som er mulig å oppnå? Egen case kan her f.eks. hentes fra en tenkt/fiktiv jobbsituasjon eller fra fritid/hobby.

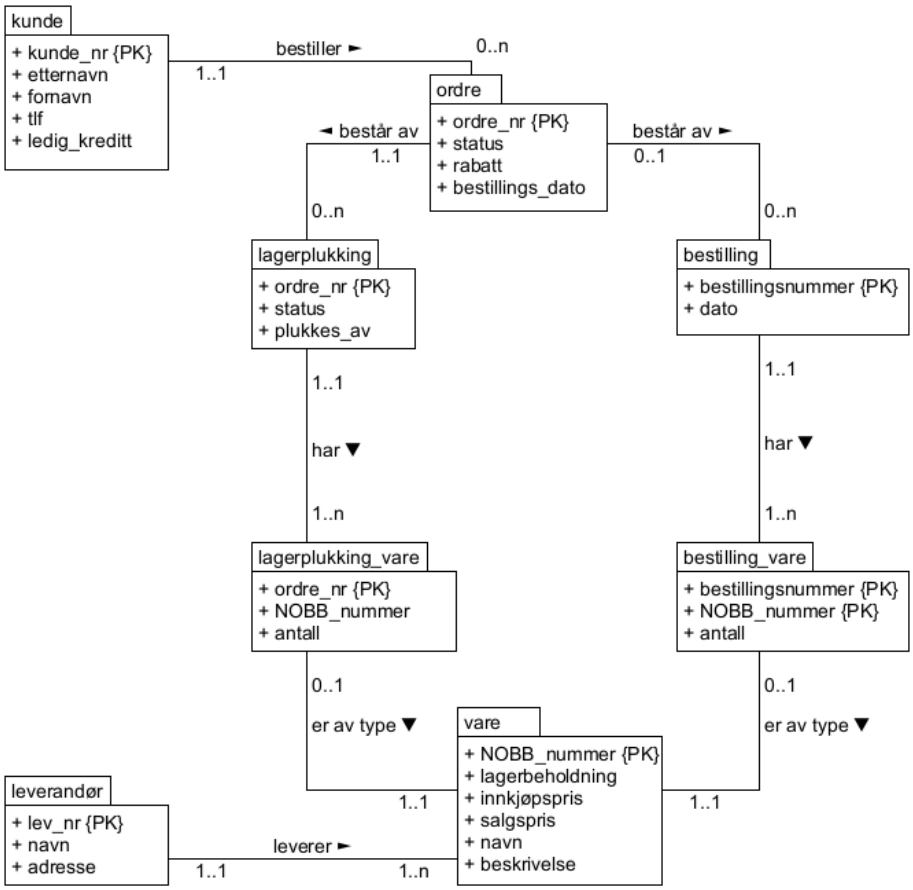
Den følgende case-en taes fra en tidligere sommerjobb hos en anonym byggevareforhandler på Vestre Rosten 97.
Den innebærer at man vil holde styr på lager beholdningen, hvilke bestillinger som kommer inn i butikken og hvilke spesialbestillinger kunder har lagt. Kunder kan også bestille varer som ligger på lager. Varene blir da gjort klare, slik at kunden bare kan komme og hente de. De kan også spesialbestille varer som ikke finnes i sortimentet, da vil varen bestilles inn fra andre anonyme byggevareforhandlere.

Behovene blir her å holde styr på hva de forskjellige kundene har kjøpt og hva de venter på å kjøpe, i tillegg til ledig kreditt. Kundene har ofte lyst til å se egen historikk, og man burde søke etter bestillinger på navn, da mange ikke har med ordrenummeret når de vil hente en ordre, selv om de får ordrenummeret på sms.

Hvis en kunde bestiller en vare som finnes på lager, så skal lagerbeholdningen selges først , varer skal kun bestilles fra andre butikker/leverandører dersom det ikke finnes på lager.
Det må skilles mellom ordre og bestillinger, da ordre er ut til kunder og bestillinger er inn til butikken. En bestilling kan være koblet opp mot en ordre. Dette betyr at bestillingen i sin helhet hører til i en ordre som en kunde har laget, og den skal derfor ikke inn i butikken. Hvis en bestilling ikke har en sammenhengende ordre så betyr det at det er en lagervare som skal selges i butikken.

Varer skal representeres med et NOBB-nummer, som er en gjennomgående standard for identifisering av varer i byggevarebransjen. Oversikt over dette kan finnes på NOBB.no. Man kan gå ut ifra at alle produkter som selges vil ha et tilhørende NOBB-nummer.

1.2.Egen relasjonsdatabase med eksempler i SQL



Relasjonene som trengs i case-en er:
kunde(kunde_nr, etternavn, fornavn, tlf, rabatt, ledig_kreditt)
ordre(ordre_nr, status, rabatt, bestillings_dato, kunde_nr*)
lagerplukking(ordre_nr*, status, plukkes_av)
lagerplukking_vare(ordre_nr*, NOBB_nummer*, antall)
bestilling(bestillingsnummer, dato, ordre_nr*)
bestilling_vare(bestillingsnummer*, NOBB_nummer*, antall)
leverandør(lev_nr, navn, adresse)
vare(NOBB_nummer, lagerbeholdning, innkjøpspris, salgspris, navn, beskrivelse, lev_nr*)

SQL spørringer: Se vedlagte filer. Vi har laget create-table setninger, disse ble brukt for å sjekke om spørringene kompilerte.

Databaseløsningen ble gjennomført i en tilfredsstillende grad. Den var relativt grei å jobbe med, og vi møtte ingen spesielle vanskeligheter når vi lagde spørringer til den. Dersom en ville ha utvidet på systemet kunne en gjerne implementert flere relasjoner mellom ansatte og salg, og ansatte og plukking. Når en har en kobling mellom en ansatt og en ordre, ville en også kunne visst hvem en skulle spørre dersom det oppstår et problem med ordren.

1.3.Løsning med XML evt. NoSQL-løsning

Høsten 2020 er det kun enkel XML som er pensum i faget. Dette står i læreboka kap. 10.8 (Semistrukturelle databaser og XML). JSON og NoSQL-løsninger er ikke pensum. Derfor inneholder denne oppgaven kun ett spørsmål om XML:Nevn kort med tekst noen fordeler evt. ulemper med å lagre alle eller noen av dataene i databasen din i oppgave 1.2 som XML.TILLEGGSOPPGAVE (IKKE PENSUM): Skriv kort om hvor godt egnet du tror en NoSQL-løsning kan være som et alternativ til relasjonsdatabasen din i oppg. 1.2

Fordeler:

- Lettere å lese data fra fil

Dersom en bruker ønsker å lese av dataene direkte fra databasen uten et grafisk grensesnitt, kan det være lettere å lese XML pga. dets innebygde struktur, med beskrivende tags så vil det være lett å skjønne hva dataene representerer.

- Plattformuavhengig

XML er tekstbasert og baserer seg på en standard definert av World Wide Consortium, noe som fører til at den databaseløsningen man utvikler både kan utvikles og leses uansett hvilket system man jobber på. Dette er derimot ikke et problem i moderne sql-løsninger heller, med mindre man bruker proprietær teknologi.

Ulemper:
XML er en løsning som bruker mange ord, så simpelthen det å skrive nye xml-filer som man skal bruke i en databaseløsning vil bli en del mer arbeid enn det å simpelthen skrive create-table setninger i mysql som er mye mer konsist.
På grunn av denne overfloden med ord som XML har så vil også en XML-løsning av en database føre til at denne databasen tar mye mer plass enn den ville ha gjort som en tradisjonell sql-løsning, da både dataene og tags-ene til hver tuppel må lagres.