## 7.1 ER Diagram

Databasen er designet med tabeller og relationer som overholder 3. normalform. Vores databasetabeller er delt op i to. En relations tabel og en del løse tabeller der indeholder forskellige måleenheder til brug for udregning af carport. Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Det er ikke alle tabellerne der overholder 3. normalform. I selve `orders`- tabellen har vi valgt at indsætte værdierne fra `measurement\_units`, `roof` og `roof\_pitch` som derfor vil give redundans. Grunden til at vælge denne metode er for at gemme kundens historik. Så hvis Fog retter eller sletter et felt i en af tabellerne, `measurement\_units`, `roof` eller `roof\_pitch` vil ændringerne ikke indvirke på kundens valg. En anden grund vil være, at en fuldt normaliseret database kunne medføre at selv de mest simple udtræk skal ske over mange joins, hvilket ofte vil være langsommere at eksekvere, end en tabel som ikke overholder alle normalformer. Hvis `orders`- tabellen skulle være uden redundans, vil man være nødt til at joine 9 tabeller og ikke kun 3 for hver gang, der skal trækkes data ud. Man vil også være nødt til at enten skille `measurement\_units`- tabellen i fire, en til hver måleenhed, eller lave en tabel der linker mellem `orders` -og `measurement\_units`- tabellen.

Tabellerne `user\_proposition` og `orders` er sat op som 1-mange relation, men bliver i vores system kun brugt som 1-1 relation. Grunden til at det kun bliver brugt som 1-1 relation er at en kunde forespørgsel er unik og bliver ikke genbrugt. Man vil i vores tilfælde sagtens kunne slå begge tabeller sammen, men vælger her at separere informationerne, så kundens personlige data som navn, adresse, telefonnummer etc. Ikke blandes med de tekniske data fra kundens indtastning. Hvis Fog på et tidspunkt vælger at bruge kundeoplysningerne til at oprette en bruger, vil det hermed være muligt at tilknytte flere ordre til `user\_proposition` tabellen.

Den eneste tabel som ikke har et automatisk generet ID er `cities`- tabellen. Det var meningen at den skulle bruges i I `user\_proposition`- tabellen, men Fog har dog valgt at postnummer og by skal være et frit skrivefelt og ikke separeres i to felter. Vi har dog tilføjet tabellen `cities`, som indeholder alle danske postnumre med dertil hørende bynavne. Hvis Fog senere hen vælger at tilføje en mere dynamisk indtastning af postnummer og bynavn, er databasen klar.

Alle tabellerne i databasen har en PRIMARY KEY som er auto increment, for nær `cities`- tabellen. Ved at bruge auto increment skaber man en unique nøgle for hver række i den enkelte tabel. Denne nøgle kan hermed bruges til at ændre, slette eller få vist en specifik række i tabellen. I vores relations tabel er der FOREIGN KEYs der refererer til PRIMARY KEYs. FOREIGN KEYs tillader krydshenvisende relaterede data på tværs af tabeller, og ved brug af CONSTRAINT, holde de relaterede data konsistente.

Da data i `user\_proposition`, `orders` og `orderline` refererer til hinanden er der tilføjet ON DELETE CASCADE og ON UPDATE CASCADE. Det vil sige at alle FOREIGN KEYs der referer til slettede også skal slettes eller alle FOREIGN KEYs der referer til ændrede skal ændres. Hvis f.eks. en Fog medarbejder sletter en kunde forespørgsel fra `user\_proposition`- tabellen, slettes også den tilhørende ordre i `orders`- tabellen og alle ordrelinjer i `orderline`- tabellen.

Tabellen `item\_list` har også en relation til `orderline`- tabellen, men her skal der hverken slettes eller ændres hvis f.eks. en kunde forespørgsel bliver slettet. For at undgå at der ændres eller slettes i `item\_list`- tabellen tilføjer vi ON DELETE RESTRICT og ON UPDATE RESTRICT.

På denne måde bliver det let at ændre og slette i tabellerne, da man kun behøver en query i stedet for tre.