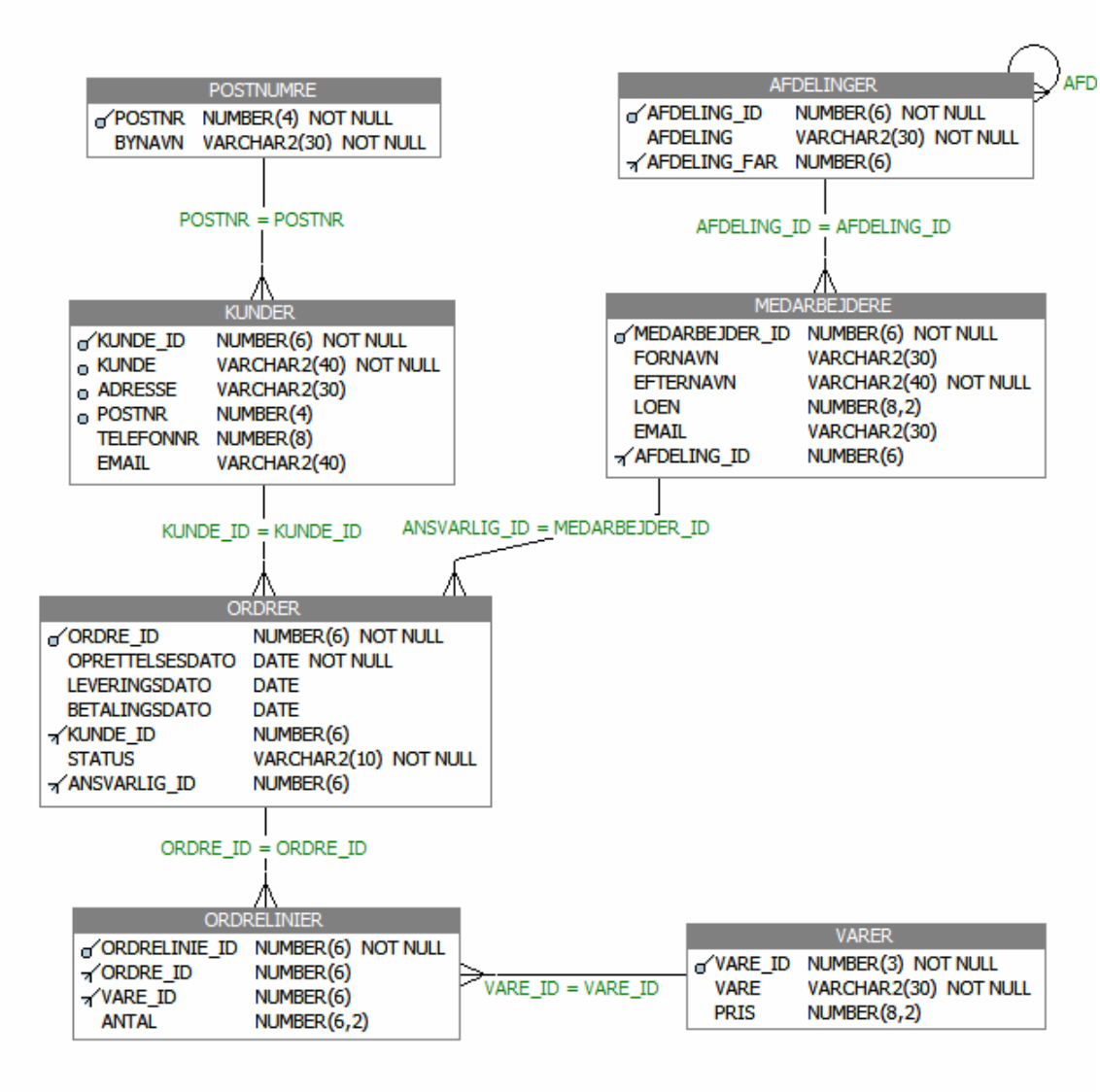
SQL-opgaver 5 – løsning

Diagrammet herunder viser, hvordan kildetabellerne gerne skal se ud efter at have løst de tidligere opgaver. Scriptet ”opgave_5.txt” indeholder også disse tabelstrukturer og alle data, som er nødvendige for at gennemføre nedenstående opgaver. Parentesen angiver hvor mange rækker, der skal returneres af forespørgslen til hver opgave, hvilke man kan bruge som tjek uden at se den vejledende løsning.



1. Vis alle kunder der er i kundetabellen. (6 rækker)

```
SELECT *
FROM kunder;
```

2. Vis alle data fra medarbejdertabellen om medarbejder med id 257. (1 række)

```
SELECT *
FROM medarbejdere
WHERE medarbejder_id = 257;
```

3. Vis kunde(-navn), telefonnr og email på alle kunder sorteret efter kundenavn. (6 rækker)

```
SELECT kunde, telefonnr, email
FROM kunder
ORDER BY kunde;
```

4. Vis kunder, hvis kundenavn indeholder et 'ø'. (Resultat lig 4 rækker)

```
SELECT *
FROM kunder
WHERE kunde LIKE '%ø%';
```

5. Vis kunder der ikke har et telefonnr. (1 række)

```
SELECT *
FROM kunder
WHERE telefonnr IS NULL;
```

6. Vis navn og pris på alle varer, der koster mellem 400 og 800 kr. (3 rækker)

```
SELECT *
FROM varer
WHERE pris BETWEEN 400 AND 800;
```

7. Lav en forespørgsel på kundenavn, adresse, postnr og bynavn. (6 rækker)

```
SELECT kunde, adresse, postnr, bynavn
FROM kunder NATURAL JOIN postnumre;
```

8. Lav en forespørgsel der viser hvilket medarbejdere der har indtastet hvilke ordrer. Følgende data skal med: Fornavn, efternavn, ordre_id og oprettelsesdato. (5 rækker)

```
SELECT fornavn, efternavn, ordre_id, oprettelsesdato
FROM medarbejdere m JOIN ordrer o ON (m.medarbejder_id =
o.ansvarlig_id);
```

9. Udvid ovenstående forespørgsel til at vise alle medarbejdere også selvom de ikke har været ansvarlig for nogle ordrer. (8 rækker)

```
SELECT fornavn, efternavn, ordre_id, oprettelsesdato
FROM medarbejdere m LEFT OUTER JOIN ordrer o ON (m.medarbejder_id =
o.ansvarlig_id);
```

10. Hvor mange rækker er der i ordrerliner? (1 række)

```
SELECT COUNT(*)
FROM ordrelinier;
```

11. Hvor mange unikke efternavne er der i medarbejdertabellen? (1 række)

```
SELECT COUNT(DISTINCT(efternavn))
FROM medarbejdere;
```

12. Vis fornavn og efternavn sammentrukket til et felt for alle medarbejdere. (6 rækker)

```
SELECT fornavn || ' ' || efternavn AS navn
FROM medarbejdere;
```

13. Vis medarbejdernes initialer (uden mellemnavne). (6 rækker)

```
SELECT substr(fornavn,1,1) || substr(efternavn,1,1) AS initialer
FROM medarbejdere;
```

14. Mogens Simonsen har været til lønforhandling, hvilket har medført at hans månedslån stiger med 8%. Gennemfør denne opdatering.

```
UPDATE medarbejdere
SET loen = loen * 1.08
WHERE fornavn = 'Mogens'
AND efternavn = 'Simonsen';
```

15. Vis varenavn og priser afrundet til hele kroner. (7 rækker)

```
SELECT vare, ROUND(pris)
FROM varer;
```

16. Vis det samlede antal enheder der er bestilt/solgt grupperet på hver ordre. (5 rækker)

```
SELECT ordre_id, SUM(antal)
FROM ordrelinier
GROUP BY ordre_id;
```

17. Lav en forespørgsel der viser alle email-adresser for kunder og medarbejdere. (9 rækker)

```
SELECT email
FROM medarbejdere
UNION
SELECT email
FROM kunder;
```

18. Lav en forespørgsel på tabellen ordrer, der udover ordre_id, oprettelsesdato og leveringsdato skal indeholder en ekstra kolonne med overskriften leveringstid. Indholdet af denne skal være en af 3 tekster: 'Langsom', 'Normal' eller 'Hurtig' afhængig af om forskellen mellem leveringsdato og oprettelsesdato er over 14 dage, mellem 7 og 14 dage eller under 7 dage. (5 rækker)

```
SELECT ordre_id, oprettelsesdato, leveringsdato,
     (CASE
     WHEN leveringsdato – oprettelsesdato < 7 THEN 'Hurtig'
     WHEN leveringsdato – oprettelsesdato > 14 THEN 'Langsom'
     ELSE 'Normal'
     END) leveringstid
FROM ordrer;
```

19. Vis navnet på den dyreste vare, vi sælger. (1 række)

```
SELECT vare
FROM varer
WHERE pris = (
  SELECT max(pris)
  FROM varer)
;
```

20. Vis navnet på de varer, som vi endnu ikke solgt nogle af. (2 rækker)

```
SELECT vare
FROM varer
WHERE vare_id IN (
  SELECT vare_id FROM varer
  MINUS
  SELECT vare_id FROM ordrelinier)
;
```

21. Lav en hierarkisk forespørgsel på tabellen afdelinger med udgangspunkt i den overordnede virksomhed. Afdeling_id, afdeling og niveau/level skal vises. (5 rækker)

```
SELECT afdeling_id, afdeling, LEVEL
FROM afdelinger
CONNECT BY PRIOR afdeling_id = afdeling_far
START WITH afdeling_far is null;
```

22. Vis kundenavn, ordrestatus og hvilke varer (varenavn, antal og pris) vi har solgt til hver kunde. Sorter resultatet efter kundenavn. (10 rækker)

```
SELECT kunde, status, vare, antal, pris
FROM kunder NATURAL JOIN ordrer NATURAL JOIN ordrelinier NATURAL
JOIN varer
ORDER BY kunde;
```

23. Tilføj en kolonne til ovenstående forespørgsel der ganger antal med pris. (10 rækker)

```
SELECT kunde, status, vare, antal, pris, antal * pris AS total
FROM kunder NATURAL JOIN ordrer NATURAL JOIN ordrelinier NATURAL
JOIN varer
ORDER BY kunde;
```

24. Opret et view kaldet kunder_varer pba. forespørgslen i ovenstående opgave.

```
CREATE VIEW kunder_varer AS
SELECT kunde, status, vare, antal, pris, antal * pris AS total
FROM kunder NATURAL JOIN ordrer NATURAL JOIN ordrelinier NATURAL
JOIN varer
ORDER BY kunde;

SELECT * FROM kunder_varer;
```

25. Vis den totale omsætning for alle de varer der er bestilt/solgt. (1 række)

```
SELECT SUM(total) AS omsaetning
FROM kunder_varer;
```