- Welke tabellen en welke kolommen
    1. Definieer voor elke tabel een primaire-sleutel
       Wanneer een tabel meerdere kandidaat-sleutels bevat, moet er een keuze worden gemaakt (Van der Lans, 1998, p.320). De primaire-sleutel is altijd de kandidaat-sleutel die uit het kleinste aantal kolommen bestaat. Dit vereenvoudigt de SELECT-instructies als tabellen gekoppeld moeten worden. Een alternatief is het kiezen van een kolom die 'interne' waarden bevat; uw organisatie is eigenaar van de waarden. Als het alternatief niet mogelijk is, moet de primaire-sleutel de kolom zijn die de minste opslagruimte verbruikt.
    2. Elke determinant in een tabel moet een kandidaat-sleutel van die tabel zijn
       Deze richtlijn wordt ook wel Boyce-Codd normaal-vorm genoemd (Van der Lans, 1998, p.320). Hierbij is kolom A determinant van kolom B als bij elke verschillende waarde in A maximaal één verschillende waarde in B hoort. Een determinant kan ook uit meerdere kolommen bestaan. Bijvoorbeeld naam + voorletters. Of kolom A een determinant is van kolom B, kan berekend worden met de volgende SELECT-instructie. Kolom A is een determinant van kolom B, als de SELECT-instructie *geen* resultaat oplevert.

       ```
       SELECT A
       FROM [table_name]
       GROUP BY A
       HAVING Count (Distinct B) > 1
       ```

       Wanneer een tabel niet aan richtlijn 2 voldoet, kunnen bepaalde feiten meerdere malen opgeslagen zijn. Dit leidt tot ingewikkelder mutaties, inefficiënt gebruik van de opslagruimte en ten slotte tot inconsistente gegevens.
    3. Plaats geen repeating groups in een tabel
       Repeating groups zijn kolommen met hetzelfde soort gegevens, dezelfde betekenis en die in dezelfde tabel zijn opgenomen (Van der Lans, 1998, p.321). Hierdoor worden instructies ingewikkelder. Dit kan opgelost worden door meerdere kolommen als primaire-sleutel te gebruiken.
    4. Voeg kolommen niet samen
       Sommige kolommen zoals STRAAT, HUISNUMMER en PLAATS kan je samenvoegen tot één kolom genaamd ADRES. Wanneer je de huisnummer of plaats wilt opvragen, moet je echter gebruik maken van expressies. Dit leidt tot slechte verwerkingstijden (Van der Lans, 1998, p.324). Daarnaast moet je voor het updaten van een straatnaam, elke rij apart bekijken en wijzigen.
- Toevoegen van redundante gegevens
    5. Voeg redundante gegevens toe indien de verwerkingssnelheid van SELECT-instructies onacceptabel traag is
       De vorige richtlijnen (1 t/m 4) maken het formuleren van SELECT- en mutatie-instructies eenvoudig. Mutatie-instructies worden snel verwerkt, maar de SELECT-instructies zijn erg traag door de hoeveelheid 'joins' om tabellen te koppelen (Van der Lans, 1998, p.327). Deze SELECT-instructies kunnen versneld worden met behulp van redundante gegevens. Het toevoegen van redundante gegevens wordt ook al denormaliseren genoemd. Een nadeel van denormaliseren is het dupliceren van gegevens, wat leidt tot dubbele opslagruimte. Een ander nadeel is dat het meer mutatie-instructies vereist. Er moet dus overwogen worden wat belangrijker is: de verwerkingstijd van SELECT-instructies, de verwerkingstijd van mutaties of de benodigde opslagruimte.
- Het kiezen van een datatype voor een kolom
    6. Gebruik dezelfde datatypes voor kolommen die met elkaar vergeleken kunnen worden
       Twee datatypes zijn hetzelfde als zowel het datatype als de gedefinieerde lengte hetzelfde zijn (Van der Lans, 1998, p.327). Sommige SQL-producten kunnen instructies met vergelijkingen tussen kolommen met verschillende datatypes wel verwerken, maar uiterst langzaam.
    7. Geef een kolom alleen een numeriek datatype als met die kolom gerekend kan worden
       Kolommen met numerieke datatypes zijn nodig als er met de waarden van de kolommen gerekend gaat worden (Van der Lans, 1998, p.328). Een numeriek datatype vergt minder

opslagruimte, maar het is belangrijk om te onthouden dat codestelsels kunnen veranderen. Een huisnummer kan bijvoorbeeld ook uit letters bestaan. Definieer een kolom dus niet numeriek als het niet per se nodig is. Conversies van numerieke naar alfanumerieke waarden zijn namelijk niet eenvoudig.

8. Definieer een kolom niet te krap
   De grootste waarde, die er mogelijk in opgeslagen kan worden, moet in de kolom passen (Van der Lans, 1998, p.328). Het is belangrijk om op eventuele toekomstige waarden te letten.
9. Gebruik het datatype VARCHAR niet voor alle alfanumerieke kolommen
   Kolommen met een alfanumeriek datatype kunnen kiezen uit twee datatypes: CHAR of VARCHAR (Van der Lans, 1998, p.328). VARCHAR is ontworpen om opslagruimte te besparen, maar is niet altijd de beste keuze. Voor elke waarde wordt de lengte ervan geregistreerd. Hierdoor vergt het meer opslagruimte. Daarnaast is VARCHAR trager dan CHAR in SELECT- en mutatie-instructies.
10. Gebruik NOT NULL als een kolom altijd in elke rij gevuld moet zijn
    NULL-waarden moeten nooit op een kunstmatige manier gebruikt worden (Van der Lans, 1998, p.329). Daarnaast moet een NULL-waarde nooit als representatie voor iets anders dan ‘waarde onbekend’ worden gebruikt.