Van een ER-diagram naar een database specificatie in SQL

Huub de Beer

Eindhoven, 4 juni 2011

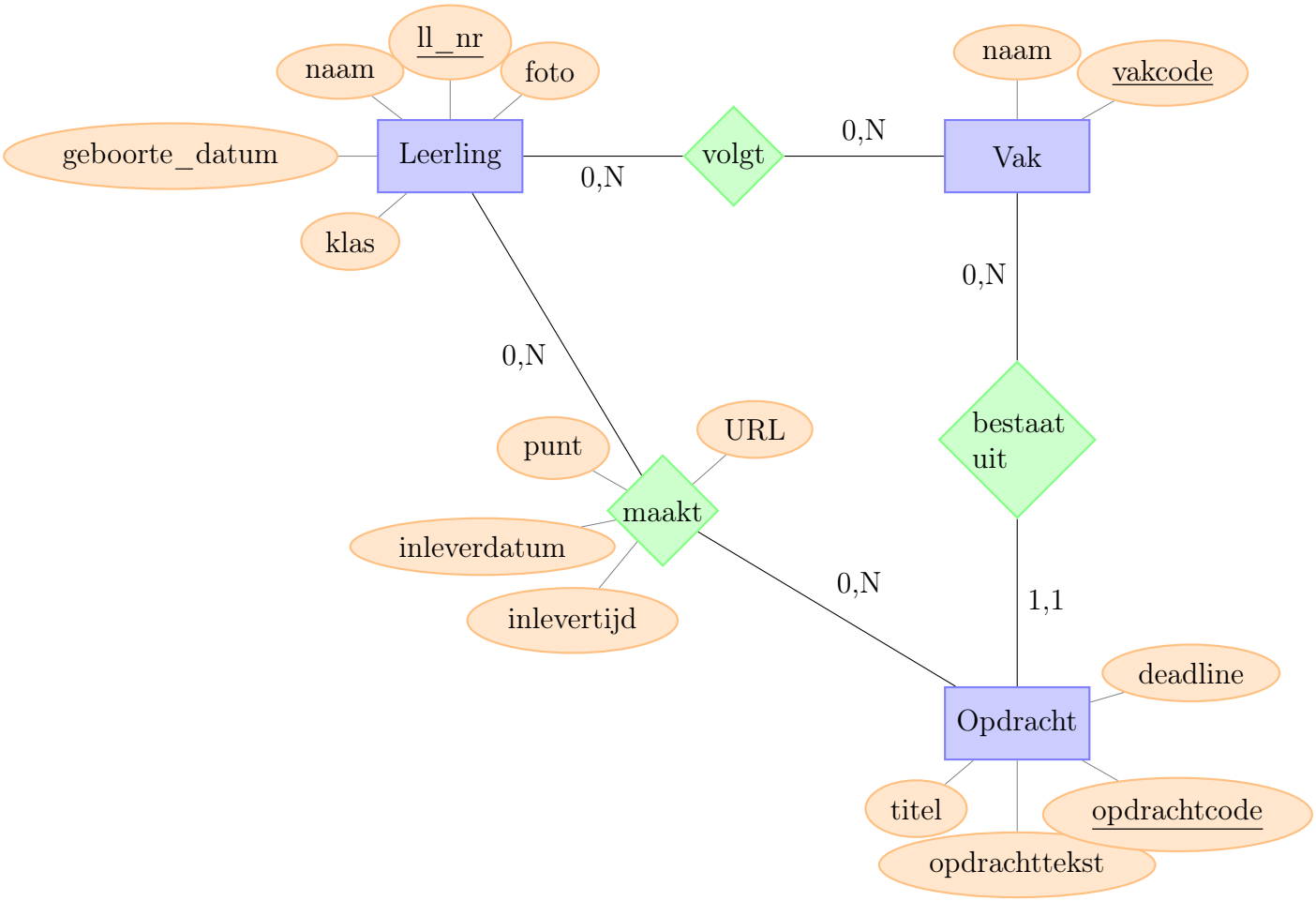
Inhoudsopgave

1 Inleiding

Het omzetten van een ER-diagram naar een database specificatie in SQL verloopt in twee stappen. Allereerst zet je het ER-diagram om naar een database specificatie in het relationele model (zie paragraaf 2). Vervolgens zet je die specificatie in het relationele model om naar een database specificatie in SQL (zie paragraaf 3). Tot slot worden er een aantal opgaven gegeven waarmee je deze tweeledige omzetting kunt oefenen.

2 Van een ER-diagram naar het relationele model

In hoofdstuk 3 van Module VIII heb je geleerd hoe je een ER-diagram omzet naar een database specificatie in het relationele model. Dit proces wordt hier



Figuur 1: Een ER diagram voor een eenvoudige ELO

nog eens kort herhaald. In figuur 1 zie je een eenvoudig ER-diagram voor een educatieve applicatie. Een vak bestaat uit een aantal opdrachten; leerlingen volgen een vak en maken opdrachten.

Je vertaalt *entiteittypen* en *attribuuttypen* in het ER-diagram direct naar, respectievelijk, *relaties* en *attributen* in het relationele model. Ook de *identiteit* van een entiteittype heeft een vergelijkbare tegenhanger in het relationele model: de *primaire sleutel* van een relatie. Het entiteittype leerling vertalen we dus als volgt naar de relatie leerling: leerling (ll_nr, naam, geboorte_datum, foto, klas). De primaire sleutel van een relatie onderstreep je; elke relatie heeft een primaire sleutel.

Het vertalen van relatietypen uit het ER-diagram is iets lastiger. Een relatietype tussen twee (of meer) entiteittypen geven we in het relationele model aan door de primaire sleutel van een relatie op te nemen in een andere

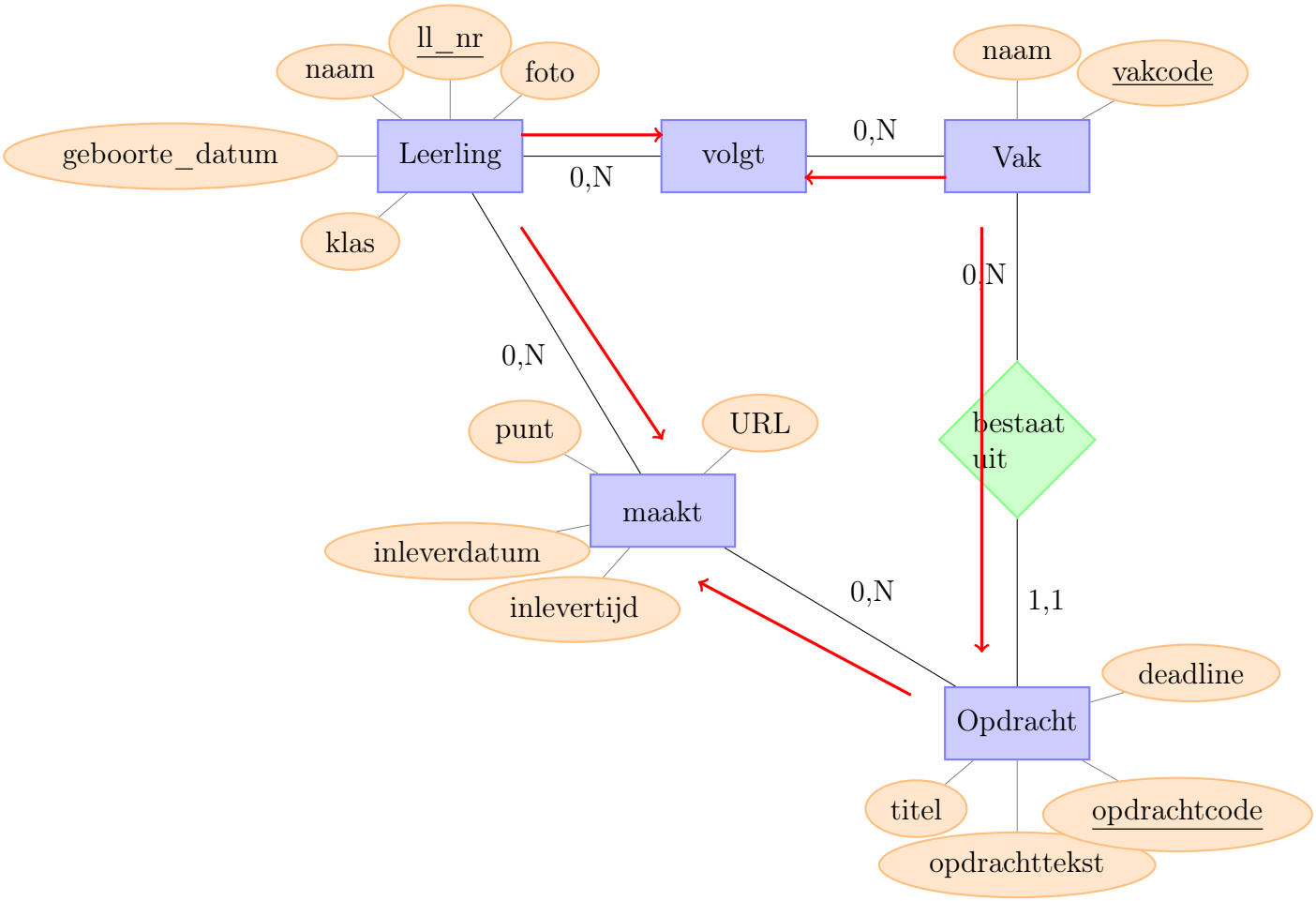
relatie als een *vreemde sleutel*. Zo'n vreemde sleutel verwijst dus naar een primaire sleutel in een andere relatie en daarmee zijn de twee relaties met elkaar verbonden. Een vreemde sleutel plaats je tussen < en >.

Er zijn drie verschillende soorten relatietypen: één-op-één, één-op-veel en veel-op-veel. Deze soorten relatietypen vertaal je als volgt naar het relationele model:

1. Bij een één-op-veel relatietype komt de primaire sleutel van het entiteittype met cardinaliteit N als een vreemde sleutel bij het entiteittype met cardinaliteit 1. We annoteren dit in het ER-diagram met een pijl van de N-kant naar de 1-kant. (Zoals het relatietype "bestaat_uit" in Figuur 2)

2. Bij een veel-op-veel relatietype vertaal je het relatietype uit het ER-diagram naar een nieuwe *extra* relatie in het relationele model. Van beide entiteittypen die door het relatietype verbonden worden, komt de primaire sleutel als een vreemde sleutel in de nieuwe relatie. We annoteren dit in het ER-diagram door het relatietype als een rechthoek te tekenen en pijlen te zetten van beide entiteittypen naar deze rechthoek in het midden. (Zoals de relatietypen "volgt" en "maakt" in Figuur 2)

3. Bij een één-op-één relatietype maak je zelf de keuze of de primaire sleutel van het ene entiteittype een vreemde sleutel wordt bij het andere, of andersom. Ook hier geef je dat aan met een pijl. (Niet in het voorbeeld.)



Figuur 2: Het ER-diagram geannoteerd met de pijlen voor de vreemde sleutels en relatietypen die tabellen worden als rechthoeken.

De volledig geannoteerde versie van het ER-diagram van de educatieve applicatie vind je in Figuur 2. Zo'n geannoteerd ER-diagram vertaal je eenvoudig naar een database specificatie in het relationele model. Alle rechthoeken worden een relatie, alle ovalen worden attributen, alle onderstreepte ovalen primaire sleutels en de pijlen geven aan welke primaire sleutels waar een vreemde sleutel worden. In Figuur 3 zie je het relationele model van het voorbeeld.

leerling (ll_nr, naam, geboorte_datum, foto, klas)
volgt (<ll_nr>, <vakcode>)
vak (vakcode, naam)
opdracht (opdrachtcode, titel, opdrachttekst, deadline, <vakcode>)
maakt (<opdrachtcode>, <ll_nr>, URL, inleverdatum, inlevertijd, punt)

Figuur 3: De database gespecificeerd in het relationele model.

Merk op dat de relatie "maakt" een primaire sleutel heeft die bestaat uit twee vreemde sleutels. Als in het voorbeeld een leerling een opdracht vaker in zou kunnen leveren, dan zou ook de inleverdatum en inlevertijd onderdeel van de primaire sleutel uit moeten maken omdat de combinatie van het leerlingnummer en de opdrachtcode dan niet meer uniek hoeft te zijn.

Let op dat bij relatietypen die omgezet worden naar een relatie er geen identiteit is aangegeven. Aan jou de taak om de primaire sleutel te definiëren. Over het algemeen bestaat de primaire sleutel van zo'n verbindingsrelatie uit de vreemde sleutels, maar soms komen daar nog enkele andere attributen bij.

Samenvattend: eerst annoteer je het ER-diagram en vervolgens stel je het relationele model van dat geannoteerde ER-diagram op.

3 Van relationeel model naar SQL

Een database specificatie in het relationele model kun je eenvoudig omzetten naar een specificatie in SQL. SQL, voluit *Structured Query Language*, is een standaard taal om met databases te communiceren. Bijna elke database en elk database management systeem (RDMS) "spreekt" een variant van SQL. Meestal implementeert zo'n databasesysteem niet alles van SQL of wordt er iets extra's aan toegevoegd, maar de basis is toch SQL. Daarom is het belangrijk dat je SQL kent: je kunt dan snel in voldoende mate met bijna elk databasesysteem overweg.

SQL bestaat uit twee delen: SQL-DDL en SQL-DML. *SQL - Data-Manipulation Language* (SQL-DML) gebruiken we om informatie in de database op te vragen, te veranderen, te verwijderen of in te voeren. SQL-DML is het onderwerp van hoofdstuk 4 van Module VIII.

SQL - Data-Definition Language (SQL-DDL) gebruiken we om nieuwe (onderdelen van) databases te specificeren. Je zet een specificatie in het relationele model om in een specificatie in SQL met behulp van SQL-DDL.

Alhoewel SQL-DDL vrij uitgebreid is, hoeven jullie maar een klein deel te kennen: jullie moeten in staat zijn om met SQL-DDL een eenvoudige tabel aan te maken.

De relatie leerling (ll_nr, naam, geboorte_datum, foto, klas) specificeren we met behulp van SQL-DDL als volgt:

```
CREATE TABLE leerling (
  ll_nr  CHAR(6) NOT NULL,
  naam VARCHAR(20) NOT NULL,
  geboorte_datum DATE NOT NULL,
  foto BLOB,
  klas  VARCHAR(5) NOT NULL,
  PRIMARY KEY (ll_nr)
);
```

Zo vertel je de database letterlijk om een tabel aan te maken met de naam "leerling" die bestaat uit de velden ll_nr, naam, geboorte_datum, foto en klas. Het veld ll_nr is bovendien de primaire sleutel. Elk veld geef je ook een passend *type*, maar daarover later meer. Met **NOT NULL** geef je aan dat een veld altijd ingevuld moet worden en de speciale waarde **NULL** dus niet is toegestaan. In dit voorbeeld hoeft blijkbaar niet elke leerling een foto te hebben. (Zonder expliciete vermelding van **NULL** of **NOT NULL** geef je aan dat de **NULL**-waarde is toegestaan.)

Een relatie met een vreemde sleutel, zoals maakt (<opdrachtcode>, <ll_nr>, URL, inleverdatum, inlevertijd, punt) vertaal je op vergelijkbare wijze, er komen enkel twee vreemde sleutel specificaties bij:

```
CREATE TABLE maakt (
  opdrachtcode INTEGER NOT NULL,
   ll_nr  CHAR(6) NOT NULL,
   url  VARCHAR(50),
  inleverdatum DATE,
   inlevertijd  TIME,
  punt NUMERIC(3,1),
  PRIMARY KEY (opdrachtcode, ll_nr),
  FOREIGN KEY (opdrachtcode) REFERENCES opdracht,
  FOREIGN KEY (ll_nr) REFERENCES leerling
);
```

Net zoals je aangeeft welke velden de primaire sleutel vormen, zo geef je aan welke velden een vreemde sleutel vormen; je geeft ook aan naar welke tabel deze vreemde sleutel verwijst. Je geeft voor elke tabel waarnaar een vreemde sleutel verwijst een aparte vreemde sleutel specificatie.

Let op dat de vreemde sleutels precies hetzelfde type hebben als de primaire sleutel waarnaar ze verwijzen. Indien een vreemde sleutel dezelfde naam heeft als een al bestaand veld in de relatie, hernoem dan de vreemde sleutel. Een naam kan in een relatie maar één keer gebruikt worden.

De rest van het voorbeeld vertaal je in de opgaven naar SQL-DDL.

Tot slot: elk SQL-query dien je met een punt-komma af te sluiten. Dat geldt ook voor "queries" in SQL-DDL ...

3.1 Typen

Bij het vertalen van een relatie naar SQL is het jouw taak om de *meest geschikte* typen voor de attributen te kiezen. SQL kent veel verschillende typen en de meeste databasesystemen hebben nog veel meer typen. Jullie hoeven alleen maar de volgende typen te kennen:

SQL naam	**betekenis**
CHAR(n)	Een stukje tekst van precies n tekens. Bruikbaar voor codes van vaste lengte
VARCHAR(n)	Een stukje tekst van maximaal n tekens. Bruikbaar voor kleine stukjes tekst waar van tevoren de lengte niet bekend is, maar in elk geval kleiner of gelijk aan n tekens. Voorbeelden: naam, adres, woonplaats, ...
TEXT	Een willekeurig lange tekst. Ongeschikt voor korte stukjes tekst zoals namen, titels, codes enzovoorts. Uitermate geschikt voor tekstbestanden, beschrijvingen, samenvattingen, enzovoorts
INTEGER	Een geheel getal. Ook geschikt voor codes
FLOAT	Een reëel getal
NUMERIC(a,k)	Een kommagetal van maximaal a cijfers, waarvan k cijfers achter de komma. Bijvoorbeeld bruikbaar in financiële toepassingen of voor punten in educatieve toepassingen
BOOLEAN	De waarde **TRUE** of **FALSE**
DATE	Een datum
TIME	Een tijd
BLOB	Binary Large OBject: bedoeld voor binaire bestanden zoals foto's, programmabestanden, films, enzovoorts

Ongeschikte typen worden fout gerekend. Kies dus geen TEXT voor een

naam, of **VARCHAR** voor een datum. Bij **CHAR** en **VARCHAR** geef je ook altijd aan uit hoeveel tekens (maximaal) een waarde in een veld mag hebben. Vergelijkbaar geef je ook bij **NUMERIC** aan uit hoeveel cijfers een getal maximaal bestaat en hoeveel cijfers daarvan achter de komma staan.

3.2 Optionaliteit en de **NULL** waarde

De cardinaliteit van een relatietype in het ER-diagram vertaal je naar het relationele model en SQL met vreemde sleutels en soms door middel van een extra relatie. Over optionaliteit hebben we tot nu toe nog niet gesproken.

Als de optionaliteit van een één-op-één of een één-op-veel relatietype 0 is, dan hoeft de vreemde sleutel geen waarde te hebben. Met andere woorden, het veld met de vreemde sleutel kan dan een **NULL** waarde bevatten. Zorg dat de SQL specificatie van de relatie dat ook toelaat!

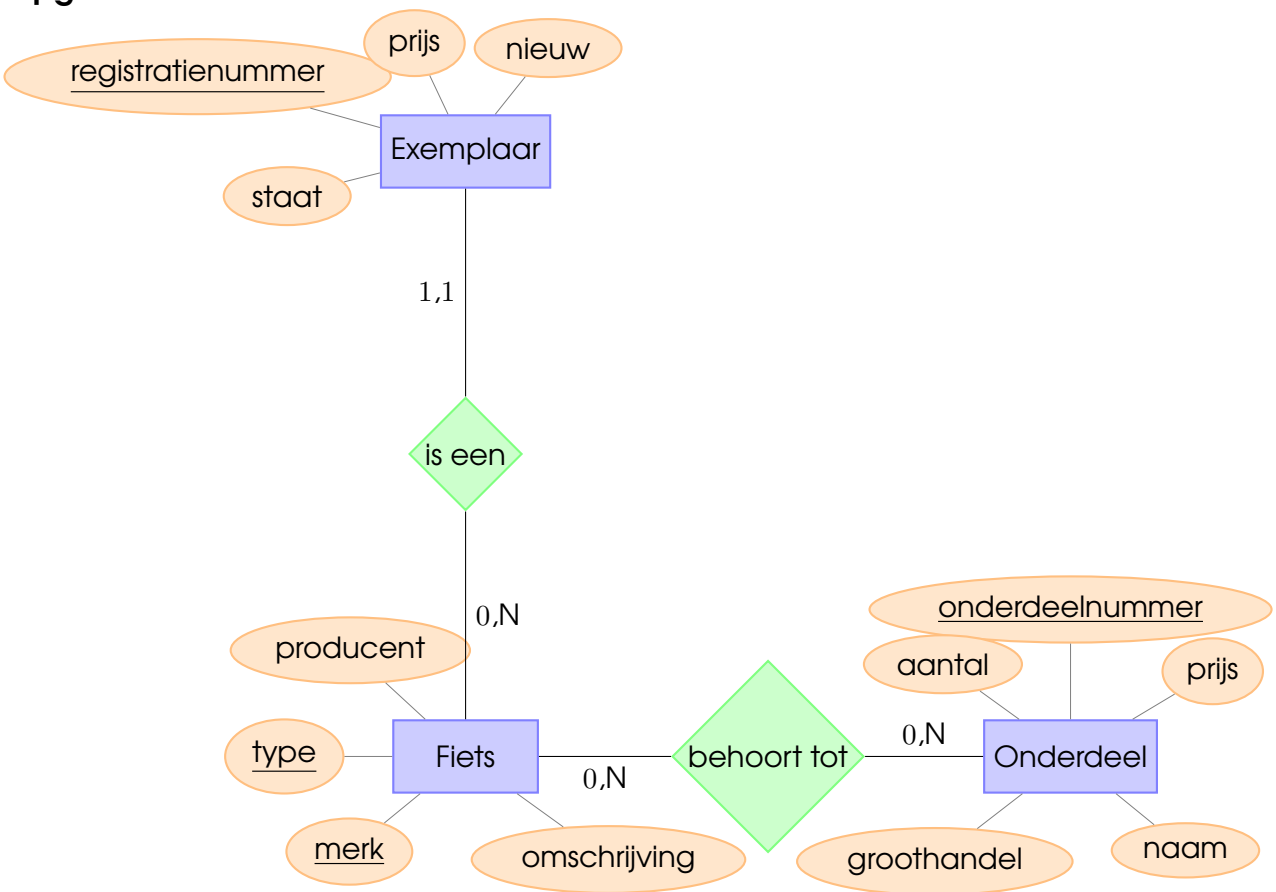
Bij een optionaliteit 1 moet een entiteit altijd met een relatie meedoen. Dat houdt in dat een vreemde sleutel nooit de **NULL** waarde mag aannemen. Daarnaast moet de referentiële integriteit gewaarborgd blijven. Wat gebeurt er als je een rij verwijdert, moeten dan alle andere rijen die naar deze rij verwijzen ook verwijderd worden? En bij het invoeren van een rij moet de rij waarnaar verwezen wordt al bestaan?

In SQL-DDL kun je verschillende van dergelijke problemen oplossen, maar dat doen we in deze cursus niet. Wij gaan er van uit dat de programmeur/-databasegebruiker zelf de referentiële integriteit in de gaten houdt.

4 Opgaven

opgave 1 Zet het relationeel model van het voorbeeld (Figuur 3, pagina 5) om naar een database specificatie in SQL. De relaties "leerling" en "maakt" zijn al vertaald. Met andere woorden, geef de **CREATE TABLE** definities voor de relaties "volgt", "vak" en "opdracht".

opgave 2 Casus fietsenhandel



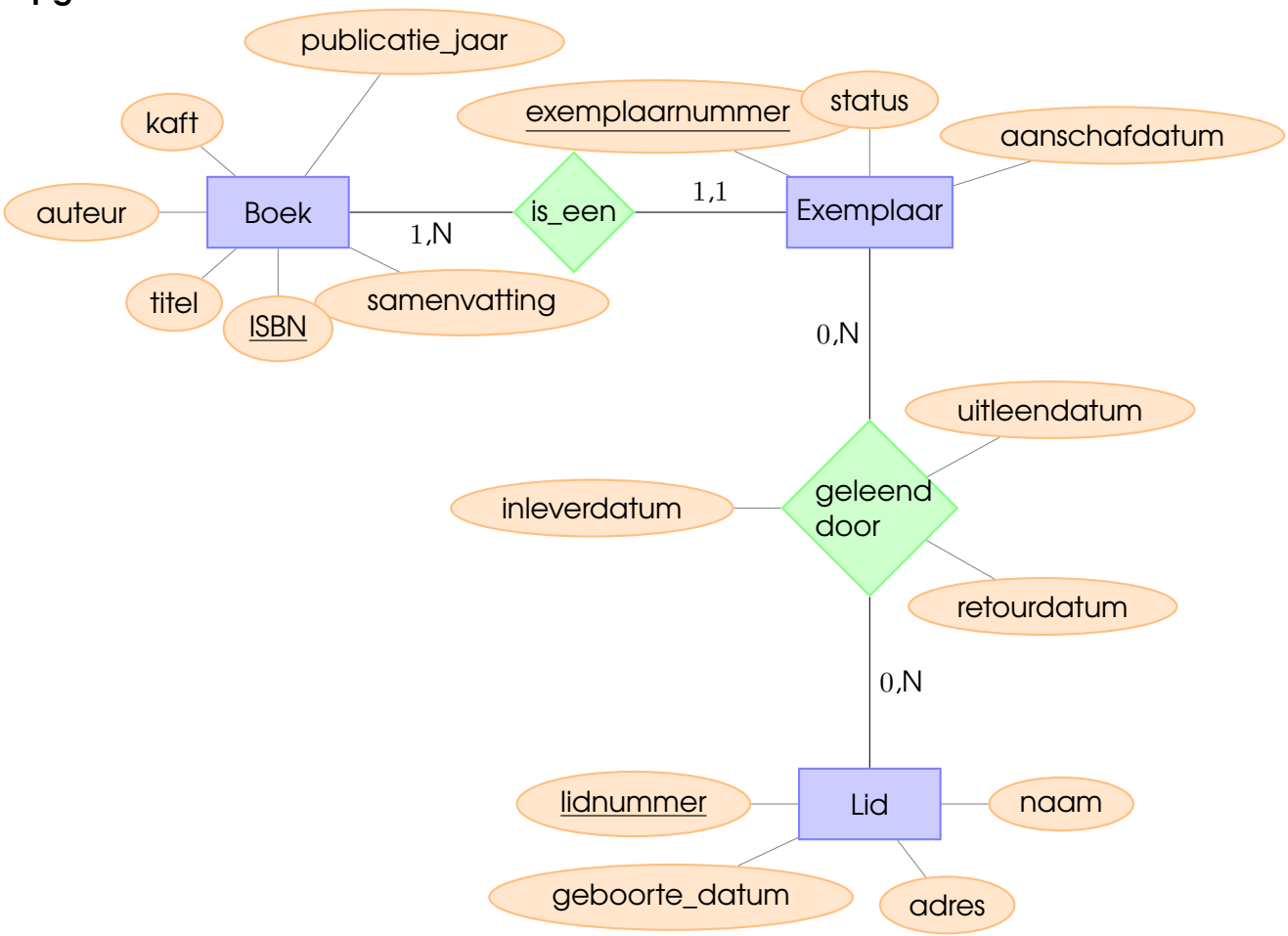
Enkele opmerkingen:

- Type, merk, naam en producent zijn korte stukjes tekst als "gazelle", "city explorer", "kids ride", enzovoorts.
- De staat is: goed, redelijk of slecht.
- De prijs heeft twee cijfers achter de komma.
- Het veld nieuw geeft aan of een fiets nieuw is of tweedehands.

Opdracht

a. Zet het ER-diagram om naar het relationele model.
b. Zet het relationele model om naar een database specificatie in SQL.

opgave 3 Casus bibliotheek



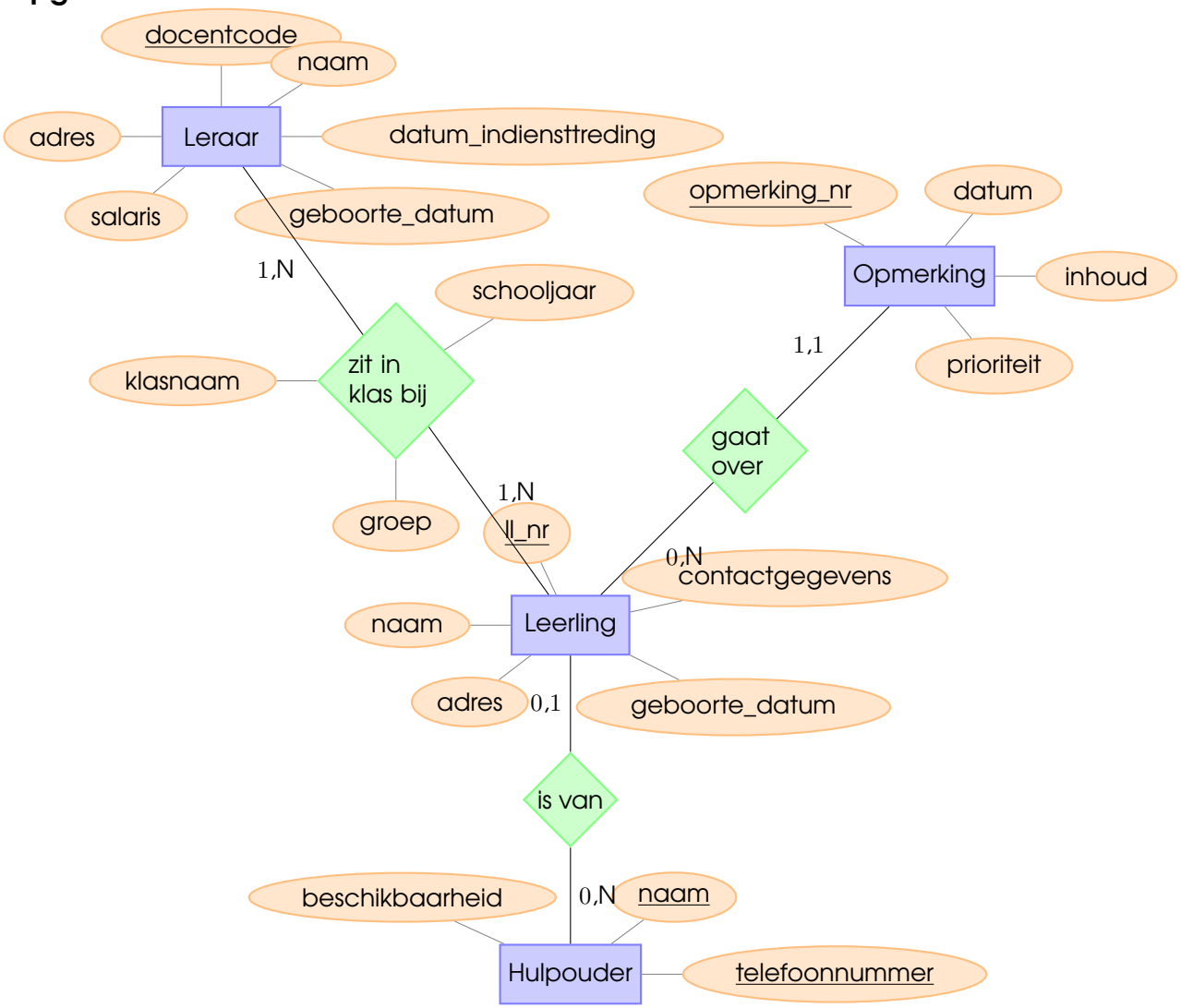
Enkele opmerkingen:

- De kaft is een foto van de voorkant van een boek.
- Het ISBN is een 10-cijferig nummer.
- De status van een exemplaar is: slecht, redelijk of goed.
- Het exemplaarnummer is een uniek geheel getal.
- Een lid kan meerdere exemplaren lenen; een exemplaar kan meerdere keren geleend worden, zelfs door hetzelfde lid.

Opdracht

a. Zet het ER-diagram om naar het relationele model.
b. Zet het relationele model om naar een database specificatie in SQL.

opgave 4 Casus basisschool



Enkele opmerkingen:

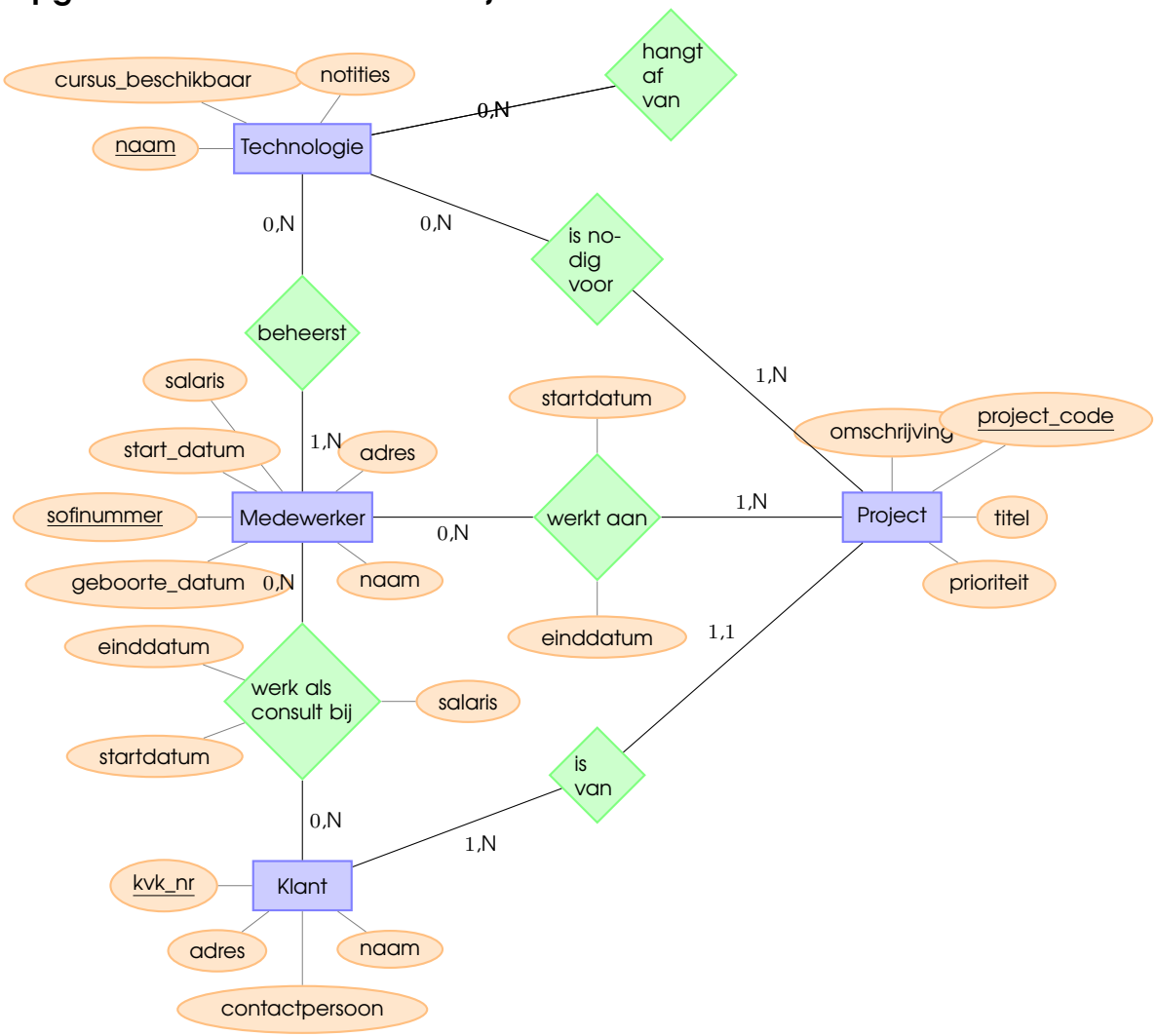
- De docentcode, het leerlingnummer en het opmerkingnummer zijn gehele getallen.
- Het salaris is een 6-cijferig getal met twee cijfers achter de komma.

- De prioriteit is: geen, laag, matig, hoog, noodgeval.
- De beschikbaarheid bestaat uit een aantal opmerkingen over wanneer en hoe de hulpouder beschikbaar is.
- De contactgegevens bestaan uit alle gegevens over de leerling waarmee contact gemaakt kan worden zoals telefoonnummers, wie de leerling verzorgt, waar de verzorgers overdag bereikbaar zijn, enzovoorts.
- De groep is 1, 2, 3, 4, 5, 6, 7 of 8.
- De klasnaam bestaat uit een cijfer en een letter.
- Het schooljaar bestaat uit twee opeenvolgende jaartallen gescheiden door een slash, bijvoorbeeld "2009/2010".
- Leerlingen zitten in hun schoolcarrière in meerdere klassen en krijgen dus les van verschillende docenten. Idem dito geven docenten aan meerdere leerlingen les per leerjaar en dat voor verschillende leerjaren.

Opdracht

a. Zet het ER-diagram om naar het relationele model.
b. Zet het relationele model om naar een database specificatie in SQL.

opgave 5 Casus softwarebedrijf



Enkele opmerkingen:

- Het sofinummer is een 9-cijferige code.
- Cursus_beschikbaar heeft de waarde **TRUE** of **FALSE**.
- Het kvk-nummer is een 12-cijferige code.
- De prioriteit is: geen, laag, normaal of hoog.
- De project_code is een geheel getal.

Opdracht

a. Zet het ER-diagram om naar het relationele model.
b. Zet het relationele model om naar een database specificatie in SQL.