Het omzetten van een ER-diagram naar SQL

Huub de Beer

Eindhoven, 4 juni 2011

Omzetting ER-diagram naar SQL in twee stappen

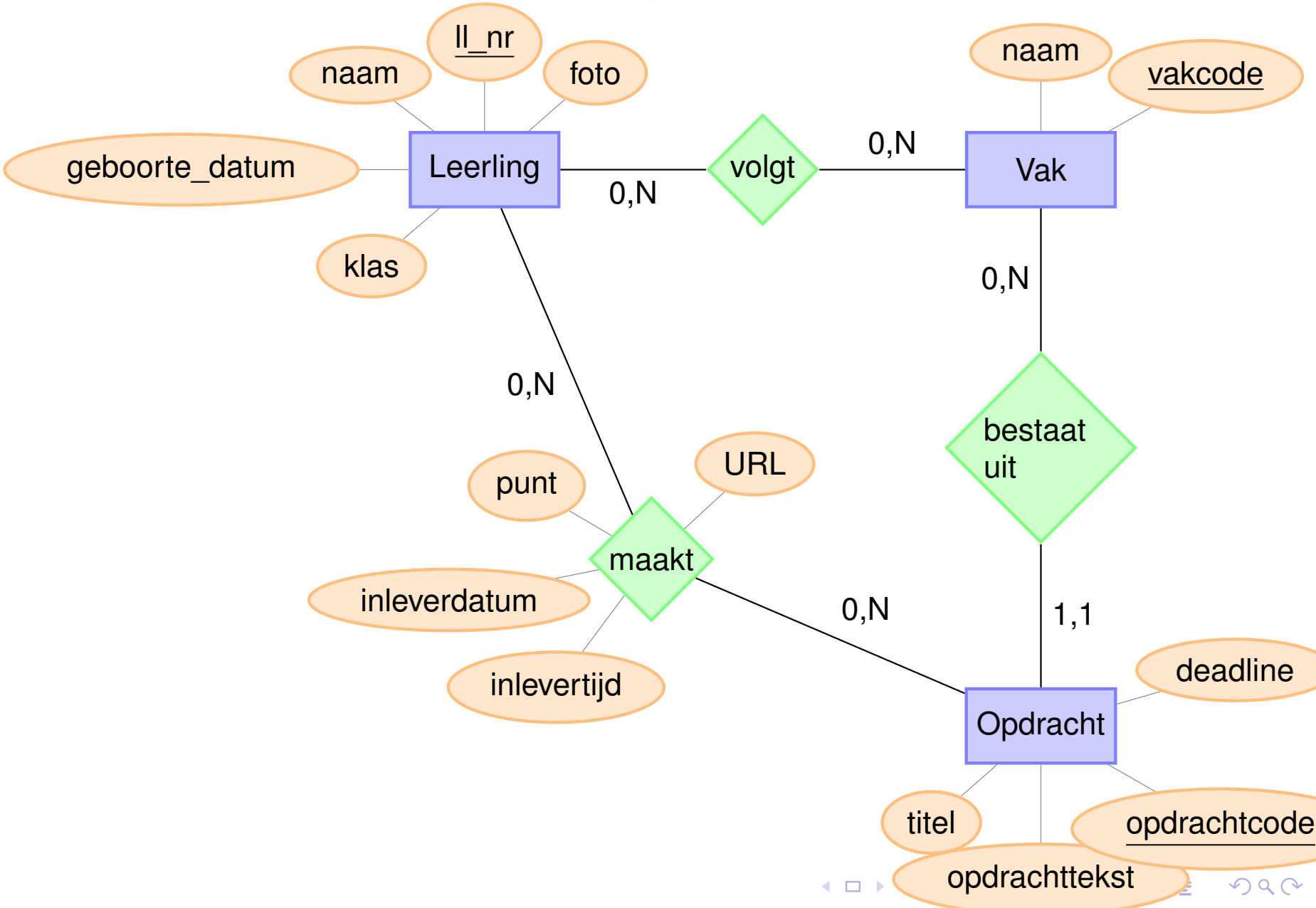
1: ER-Diagram $\rightarrow$ relationeel model

- Onderwerp van hoofdstuk 3
- Entiteittype $\rightarrow$ relatie, attribuuttype $\rightarrow$ attribuut, identiteit $\rightarrow$ primaire sleutel
- Annoteer het ER-diagram:
  - Pijlen geven aan welke primaire sleutels waar vreemde sleutels worden
  - Relatietypen met twee inkomende pijlen worden ook relaties: maak er een rechthoek van
- Uit het geannoteerde ER-diagram volgt als het ware het relationele model

2: Relationeeel model $\rightarrow$ SQL

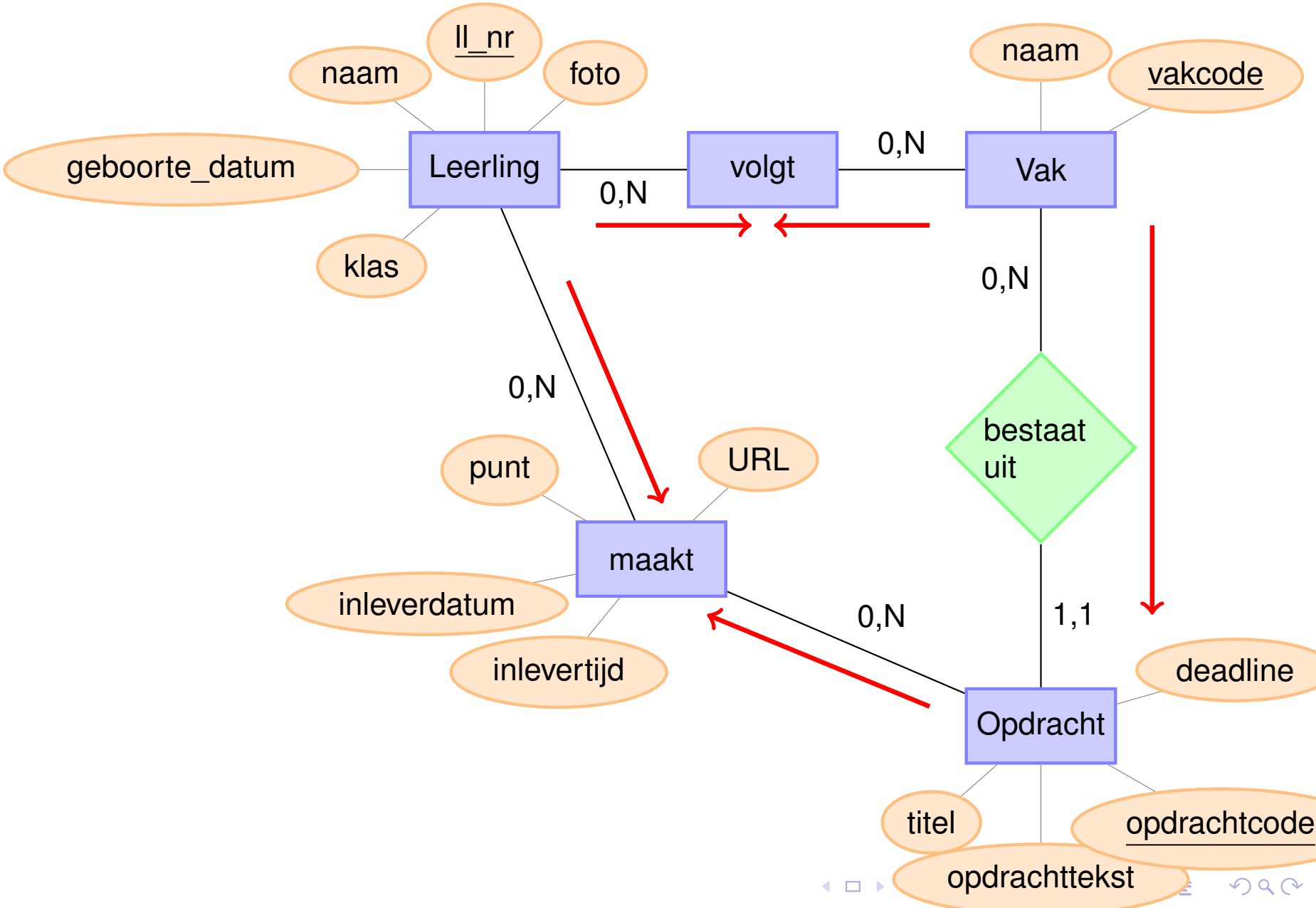
1: ERD naar relationeel model (voorbeeld/herhaling)

ER-diagram van een educatief programma



1: ERD naar relationeel model (voorbeeld/herhaling)

Geannoteerd ER-diagram



1: ERD naar relationeel model (voorbeeld/herhaling)

Relationeel model

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)

2: relationeel model naar SQL: SQL?

Structured Query Language (SQL)

- Data-Manipulation Language om informatie in de database op te vragen, veranderen, verwijderen, in te voeren (*hoofdstuk 4*)
- Data-Definition Language om een database te specificeren:
  - Jullie hoeven nu alleen maar tabellen te kunnen specificeren in SQL

Waarom SQL?

- Het is een *standaard*: elke database “spreekt” SQL
- Alle databases net iets anders: leer SQL en je kunt alle databases gebruiken
- Programmatisch altijd m.b.v. SQL met een database spreken
- Deelonderwerp TOETS

2: relationeel model naar SQL: **CREATE TABLE**

specificatie in relationeel model

leerling (ll_nr, naam, geboorte_datum, foto, klas)

specificatie in SQL

```
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)
);
```

Let op haakjes, komma's en de puntkomma: -1, -1, -1, -1, ...
Je communiceert immers in een *formele taal* met een computer! (net zoals bij programmeren)

2: relationeel model naar SQL: CREATE TABLE

specificatie in relationeel model

leerling (ll_nr, naam, geboorte_datum, foto, klas)

specificatie in SQL

```
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)
);
```

Let op haakjes, komma's en de puntkomma: -1, -1, -1, -1, ...

Je communiceert immers in een *formele taal* met een computer! (net zoals bij programmeren)

2: relationeel model naar SQL: CREATE TABLE

specificatie in relationeel model

maakt (<u><opdrachtcode>, <ll_nr></u>, URL, inleverdatum, inlevertijd, punt)

specificatie in SQL

```
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
);
```

2: relationeel model naar SQL: typen

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 naam, adres, woonplaats, . . .
TEXT	• Een willekeurig lange tekst. Uitermate geschikt voor beschrijvingen, samenvattingen, enzovoorts.
INTEGER / **INT**	• Een geheel getal. Ook geschikt voor codes.
FLOAT	• Een reëel getal.
DECIMAL(a,k) / **NUMERIC**(a,k)	• Een kommagetal van maximaal *a* cijfers, waarvan *k* cijfers achter de komma.
BOOLEAN / **BOOL**	• De waarde TRUE of FALSE
DATE	• Een datum.
TIME	• Een tijd.
BLOB	• Binary Large OBject: bedoeld voor foto's en dergelijke.

2: relationeel model naar SQL: optionaliteit en **NULL**

Vertalen van optionaliteit (één-op-veel, één-op-één)

- Optionaliteit 1: een vreemde sleutel moet altijd een (juiste) waarde hebben. *Geen* **NULL** waarde toegestaan.
- Optionaliteit 0: een vreemde sleutel *hoeft niet altijd een waarde te hebben*. **NULL** toegestaan.
- Andere beperkingen: kan in SQL, voor jullie: taak van de programmeur.

NULL

- Gebruik de clausule **NOT NULL** achter het type van een veld specificatie in SQL om aan te geven dat de **NULL** waarde niet voor mag komen.
- Sommige velden, waaronder dus sommige vreemde sleutels, mogen een **NULL** waarde bevatten.

Nu + huiswerk

Nu

- Extra lesstof op Eckartnet en zo dadelijk uitgedeeld.
- Vertaal het relationele model van de praktijkopdracht naar SQL.
- Controleer je SQL code door deze in PHPMyAdmin in te voeren.

Huiswerk

- Lees de lesstof, eerste deel is een herhaling van H3
- Maak opgaven 1 van extra lesstof: werk het voorbeeld verder uit