KU LEUVEN

Inleiding tot databanken

2. Entiteit-Relatie (ER) Model

Prof. dr. Paolo Pilozzi

Overzicht

2.1 Entiteit-Relatie (ER) Model
2.2 ER Model naar Relationeel Model

2.1 ER Model

Hoog-niveau (conceptuele) modellen
Hoofdstuk 3 – Oefenzitting 1

* Inleiding
* Ontwerp
* Conceptueel datamodel
    - ER Model
    - ER Diagramma

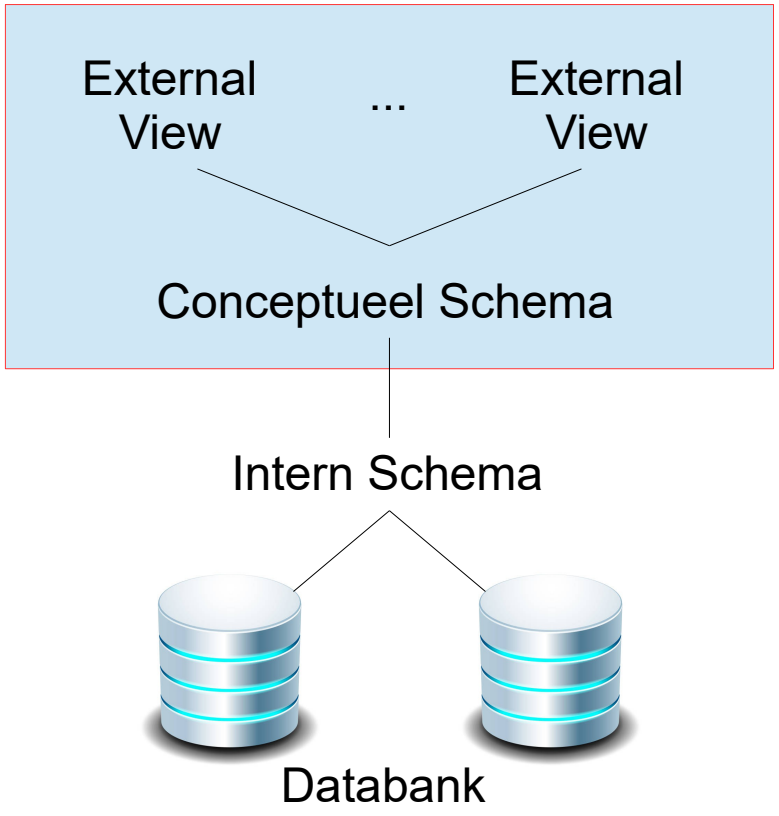
Inleiding

Cfr. 3-Schema DBMS architectuur:
Niveaus van abstractie!

- Extern: hoe gegevens tonen aan gebruikers
- **Conceptueel**: implementatiemodel
- Intern: fysische opslag en toegangspaden

Conceptueel modelleren v. databank is belangrijk(st)e fase bij ontwerp!

- Entiteit-Relatie (ER) Model als werktuig
- Onafhankelijk van (R)DBMS door abstractie
- In de praktijk:
  - Universal Modelling Language (UML)
  - Nooit volledig los van (R)DBMS



(ANSI/SPARC, 1975)

Ontwerp – Analytische fase

Analyse v. Benodigdheden (Behoeften)

= Uitgaande van gebruikersgroepen & mini-wereld, **documenteren** van:

* Wat voor data? = **Data Benodigdheden**
* Hoe werken met data? = **Functionele Benodigdheden**

Aanpassen, opzoeken, ..., m.a.w. de interacties met data

(Zie software ontwerp & tools: data-flow, sequences, scenarios, etc.)

Wees in deze fase zo volledig en overzichtelijk mogelijk.

= **uitgangspunt ontwerp!**

Itereer over gebruikersgroepen:

Overlap vermijden, Vergeetachtigheden, Ideeën geven, ...

Ontwerp – Conceptuele fase

Conceptueel ontwerp

= Data benodigdheden omzetten **naar Conceptueel schema** via hoog-niveau conceptueel data model

=> Beknopte beschrijving v/d data benodigdheden v. gebruikers:
entiteitstypes, verbanden, beperkingen, ...
=> Geen implementatiedetails ("niet-technisch").
Terugkoppelen naar gebruiker (ter referentie, verificatie)!
=> Focus op data-eigenschappen en niet dataopslag.

Ga na of aan functionele benodigdheden kan voldaan worden!

Ontwerp – Implementatie fase

Conceptueel ontwerp omzetten naar DBMS implementatie
= hoog-niveau **conceptueel** datamodel **naar implementatie datamodel**

* Afankelijk van DBMS technologie
  => Bij RDBMS: Naar relationeel datamodel
* Wordt logisch ontwerp of datamodel-mapping genoemd
* Kan (grotendeels) algoritmisch

Ontwerp – Fysiek-ontwerp fase

Niveau van interne schema's

* Opslagstructuren, toegangspaden, betandsorganisatie, etc.

Ontwerp – Itereren over fases

Implementatiemodel beïnvloedt conceptueel model

Vb. Samengestelde sleutels

Verwijzen vanuit (verschillende) tabellen naar tabel met (informatieve) samengestelde sleutel kan vaak efficiënter door toevoegen van enkelvoudige sleutel (= Trade-off)!

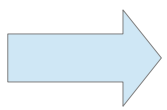
Vendor ID (integer) + Device ID (char string) zijn informatief en samengesteld uniek. Beide herhalen in verwijzende tabel(len) is meestal inefficient omwille v. redundantie. Beter is ID (integer) toevoegen en daarnaar te verwijzen.

DEVICE

VID	DID	Name
1	0001	Server
1	0002	Server
2	0001	Server
2	A001	Workstation

LINK

Src_VID	Src_DID	Src_VID	Tgt_DID
1	0001	1	0002
1	0001	2	0001
2	0001	2	A001



DEVICE

ID	VID	DID	Name
1	1	0001	Server
2	1	0002	Server
3	2	0001	Server
4	2	A001	Workstation

LINK

Src_ID	Tgt_ID
1	2
1	3
3	4

Ontwerp – Itereren over fases

Fysieke ontwerp beïnvloedt implementatiemodel

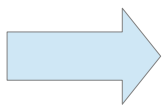
Vb. Toegangspaden (Indexering)

Op verschillende manieren zoeken naar data kan vaak efficiënter door redundante datavoorstellingen toe te voegen in tabellen (= trade-off)!

Zoeken op basis van initialen kan door gebruik te maken van de naam, maar vergt verwerking van de naam. Kan sneller door initialen explicitiet toe te voegen met een extra datakost die het waard kan zijn.

EMPLOYEE

ID	Name
1	Luc Sels
2	Peter Van Puyvelde
3	Stefan Vandewalle



EMPLOYEE

ID	Name	Initials
1	Luc Sels	LS
2	Peter Van Puyvelde	PVP
3	Stefan Vandewalle	SV

Conceptueel Datamodel

Doel: Relevant deel van werkelijkheid beschrijven
=> Objecten & Feiten

* Werkelijkheid is de mini-wereld
* Beschrijving is schema (gebruikersspecifiek)
* Via abstractie (weglaten van details)

Waarom?

* Korter bij realiteit dan implementatiemodel.
* Wat en waarom, minder hoe!
* Weglaten implementatiespecifieke details.
* Nauw verband met implementatiemodel (cfr. Mapping).

Conceptueel Datamodel

Gebruikt model: Entiteit-Relatie (ER) Model

- Entiteiten: Objecten, Acteurs
- Relaties: Verbanden tussen objecten
- Attributen: Eigenschappen
- Domeinen: Waardenverzamelingen

Kernbegrippen

- Entiteiten behoren tot Entiteitstypes
- Relaties behoren tot Relatietypes
- Waarden van Attributen behoren tot domeinen
- Sleutels: Identificeren Entiteiten en Relaties eenduidig

Doorlopend Voorbeeld: BEDRIJF

Van **data benodigdheden** naar conceptueel schemaontwerp op basis van geïntroduceerde ER model concepten.

Beschrijving Mini-wereld:

* Bijhouden v. *Werknemers*, *Departementen, Projecten en Ten Laste*
* Bedrijf georganizeerd in Departementen:
  - Departement heeft unieke Naam & uniek Nummer.
  - Departement heeft Manager (sinds Start Datum).
  - Departement kan meerdere Locaties hebben (≥1 Locaties).
* Departement controleert Projecten:
  - Projecten hebben een unieke Naam, uniek Nummer en 1 Locatie.

Doorlopend Voorbeeld: BEDRIJF

Beschrijving Mini-wereld:

* Bedrijf heeft Werknemers:
  - Werknemer heeft een Naam, een (uniek) Rijksregisternummer (RRN), een Adres, een Salaris en een Geboortedatum. Heeft Overste.
  - Werknemer is toegekend aan 1 Departement, maar mag op meerdere Projecten werken over Departementen heen.
  - Werknemer besteedt aantal uren per week aan Projecten.
* Werknemers hebben Personen ten laste:
  - Personen ten laste hebben een Naam, Geboortedatum en Verhouden zich op een bepaalde manier tot een Werknemer.

Entiteiten en hun Attributen

Het basis object in een ER model is een Entiteit.

= fysiek (vb. Werknemer) of conceptueel (vb. Departement) object

Entiteit heeft Attributen.

= eigenschappen van het object dat het object (voldoende) bepalen

vb. Werknemer heeft een Naam, RRN, Adres, Salaris, etc.

Entiteit heeft een Waarde voor elk Attribuut.

vb. Werknemer attributen: Naam, RRN, Adres, Salaris; met waarden:
"Jan Smits", "90.05.12-221.14", "Kerkstraat 14, 3000 Leuven", "2150"

Attribuut heeft een Domein (= mogelijke waarden)

* Mogelijks NULL waarde in domein.

= Waarde ongekend, niet beschikbaar, of niet van toepassing.

Entiteiten en hun Attributen

Verschillende soorten Attributen in ER Model:

* Enkelvoudig (vb. Salaris) of atomair vs. Samengesteld (vb. RRN)
  - Samengesteld kan hiërarchisch (vb. Adres – Straatadres – Straat + Nr.).
* Eénwaardig (vb. Salaris) vs. Meerwaardig (vb. Lijst van hobby's).
  - Meerwaardig mogelijks met onder/boven limiet.
* Expliciet voorgesteld (vb. RRN) vs. Afgeleid (vb. Leeftijd, Geslacht)
  - Leeftijd en geslacht kan van RRN afgeleid worden (omgekeerd niet)
  - Vb. RRN: 90.05.12-221.14
    - Geboorte datum 12 mei 1990 => We kennen Leeftijd
    - Volgnummer 221 => Oneven, dus Man (Vrouw even)
    - Controle: 900512221 mod 97 = 14
* Complex: Nesten v. Samengestelde en Meerwaardige attributen
    - vb. RRN Buren: {(90.05.12,221,14),(87.02.02,110,75)}

Ontwerp ER Schema (eerste stappen)

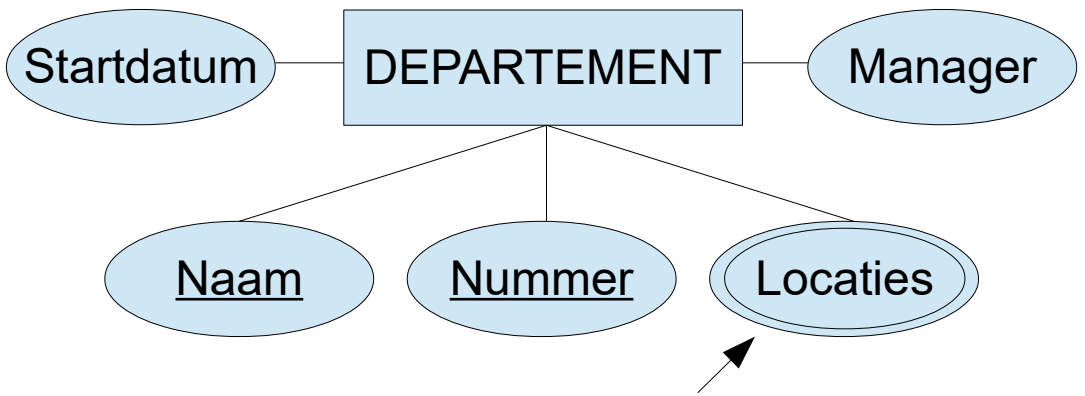
= Opstellen ER model via ER diagramma:

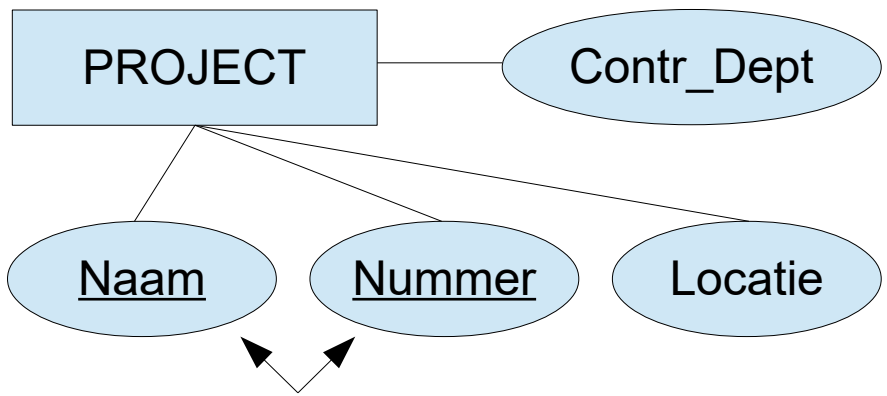
* Identificeren van **Entiteitstypes**: groepen van gelijkaardige entiteiten
  - Stellen verzamelingen van mogelijke entiteiten voor
* Entiteitstype heeft een **naam** en **attributen**
  - Elke entiteit behoort tot een entiteitstype en
  - heeft specifieke waarden voor de attributen van dat type
  - waaronder (vaak) **sleutelattributen** die de entiteiten uniek bepalen
* In **ER diagramma**: Entiteitstype als rechthoek met naam,
  daaraan verbonden ovalen met attribuutnaam,
  waarbij sleutelattributen onderlijnd worden.
  Waardenverzameling (domeinen) van attributen niet in diagramma!

Doorlopend Voorbeeld: BEDRIJF

Bedrijf georganizeerd in Departementen:
- Departement heeft unieke Naam & uniek Nummer.
- Departement heeft Manager (sinds Start Datum).
- Departement kan meerdere Locaties hebben (≥1 Locaties).

Departement controleert Projecten:
- Projecten hebben een unieke Naam, uniek Nummer en 1 Locatie.

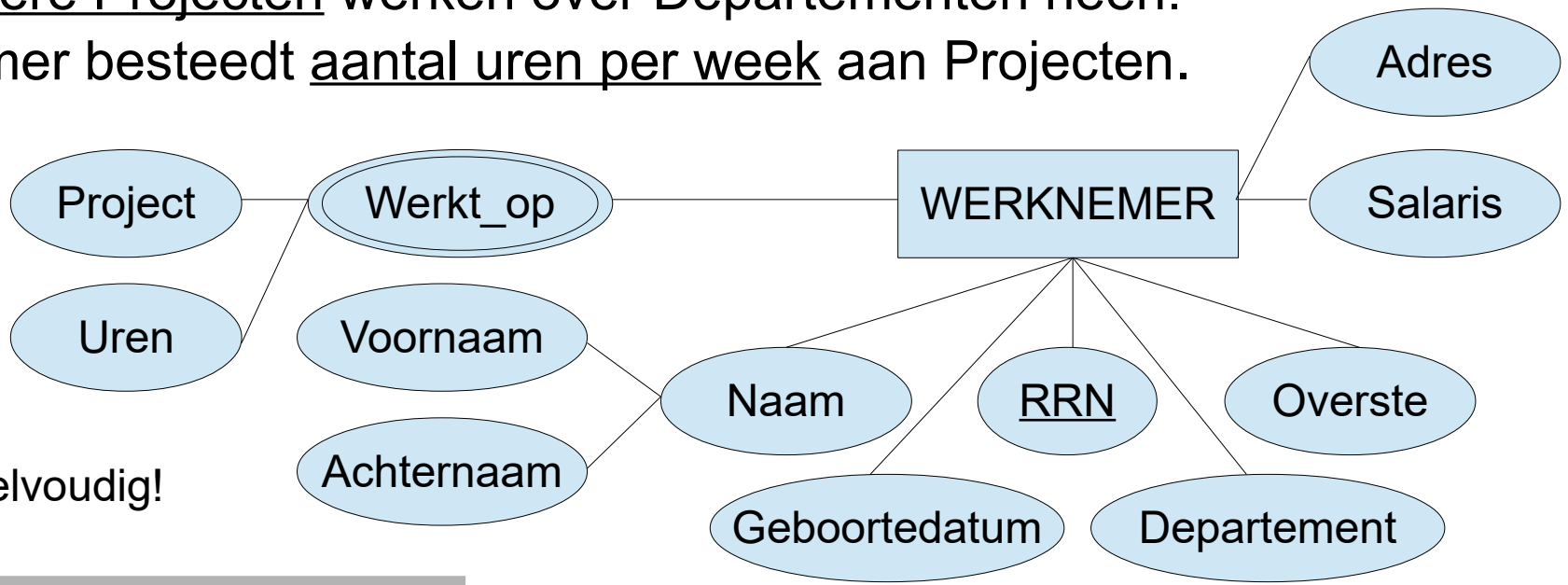




Doorlopend Voorbeeld: BEDRIJF

Bedrijf heeft Werknemers:

- Werknemer heeft een Naam, een (uniek) Rijksregisternummer (RRN), een Adres, een Salaris, en een Geboortedatum. Heeft Overste.
- Werknemer is toegekend aan 1 Departement, maar mag op meerdere Projecten werken over Departementen heen.
- Werknemer besteedt aantal uren per week aan Projecten.



Opgelet:
RRN hier enkelvoudig!

Doorlopend Voorbeeld: BEDRIJF

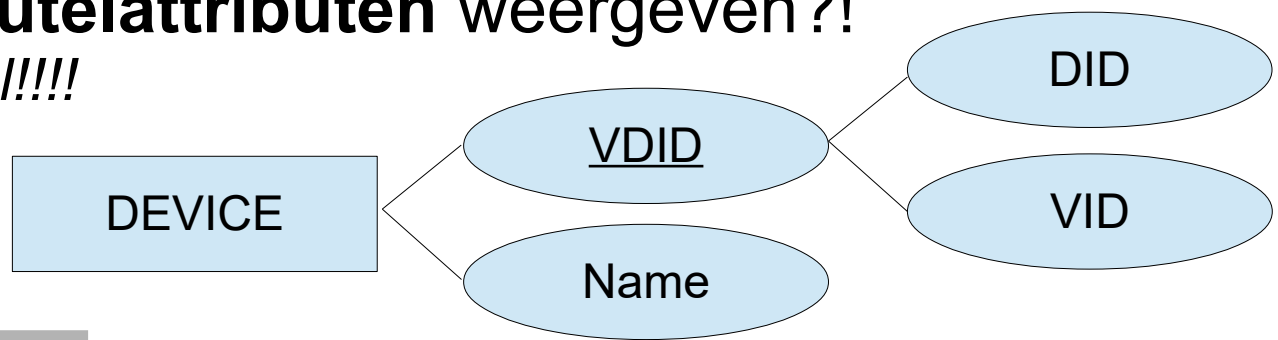
Werknemers hebben Personen ten laste:

- Personen ten laste hebben een Naam, Geboortedatum en Verhouden Zich op een bepaalde manier tot een Werknemer.

= Zwak entiteitstype (Dubbele omlijning)
=> Partiële sleutel (Stippellijn)
=> Later meer hierover

Hoe **samengestelde sleutelattributen** weergeven?!

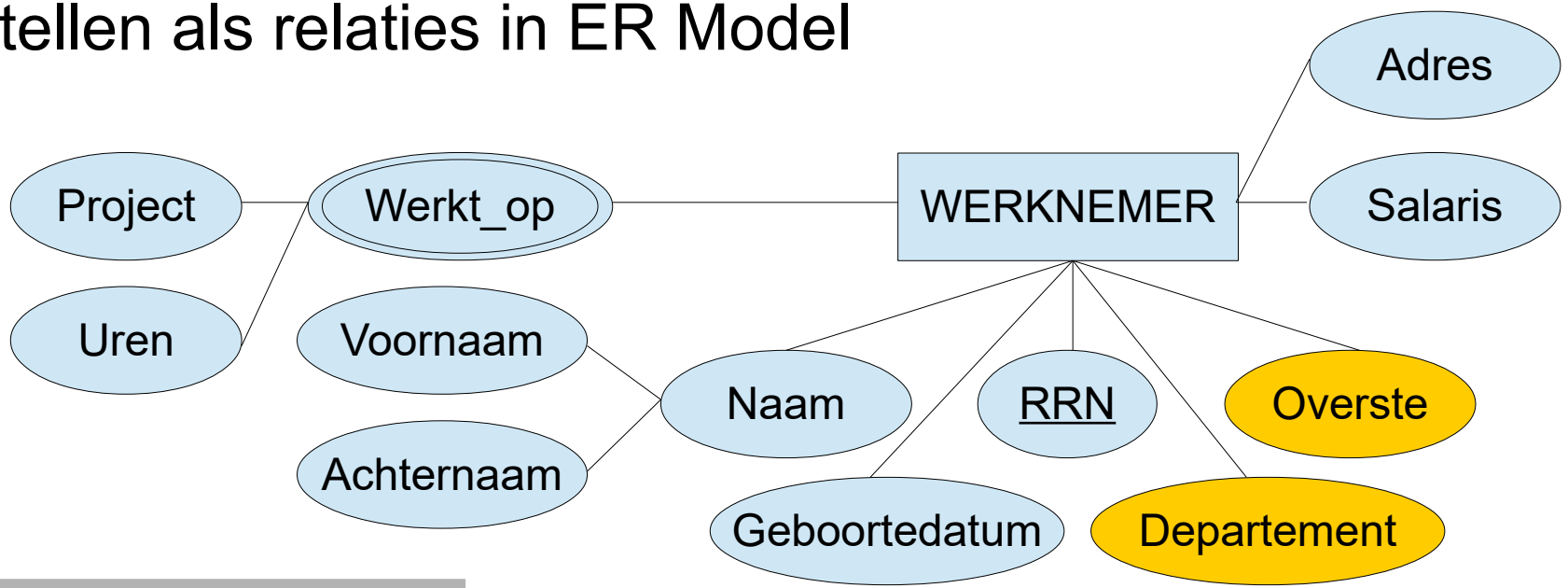
Verschil met relationeel model!!!!



Relaties in ER Model

In doorlopend voorbeeld: verschillende impliciete relaties

- Wanneer een attribuut verwijst/refereert naar een entiteit
  - vb. Departement in WERKNEMER verwijst naar DEPARTEMENT
  - vb. Overste in WERKNEMER verwijst naar WERKNEMER
- dit voorstellen als relaties in ER Model



Relaties in ER Model

Relatietype over entiteitstypes $E_1,...,E_n$

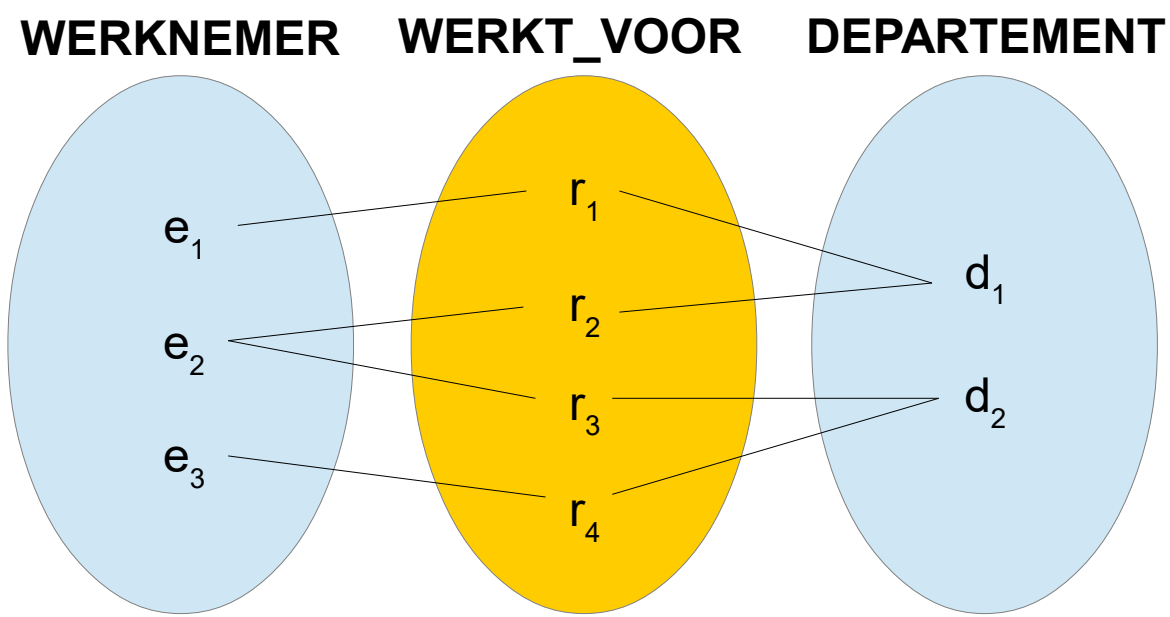
definieert verzameling verbanden over de entiteiten van deze entiteitstypes

Elke relatie (verband) associeert exact 1 entiteit per entiteitstype in het relatietype.

Relatietype WERKT_VOOR tussen entiteitstypes WERKNEMER en DEPARTEMENT met verbanden:

- e1 en e2 werken voor d1 (r1 en r2 resp.)
- e2 en e3 werken voor d2 (r3 en r4 resp.)

IN **ER DIAGRAMMA**: Ruit met naam



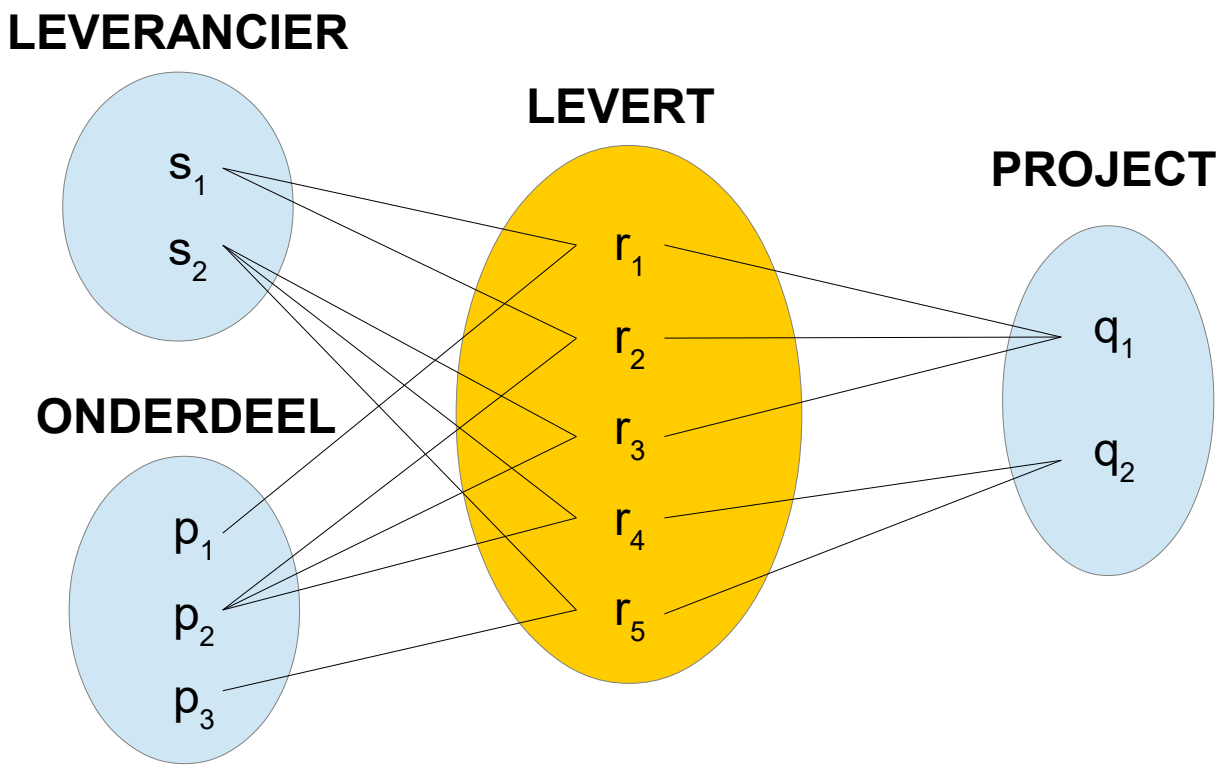
Relaties in ER Model

Relatietype heeft een **graad n** over entiteitstypes $E_1,...,E_n$

vb. binair, ternair, quaternair, etc.

WERKT_VOOR is binair relatietype (graad 2)

SUPPLY is een ternair Relatietype (graad 3)

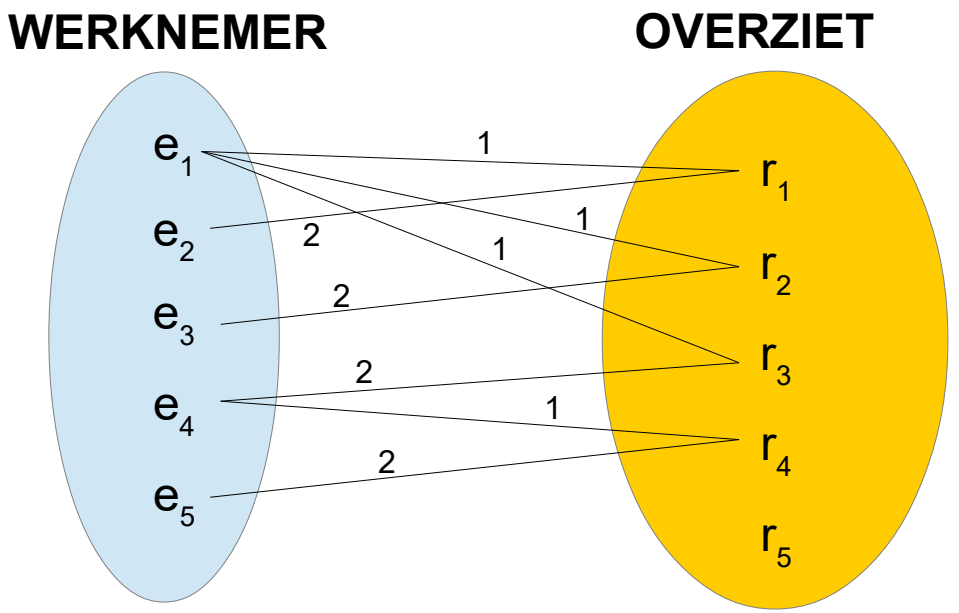


Relaties in ER Model

Entiteitstype t.a.v. een relatietype speelt altijd een rol.
Duidelijk wanneer relatietype gaat over verschillende entiteitstypes.
- D.w.z. Elke entiteit speelt duidelijke rol in de relatie.

Bij **recursief relatietype** is dat niet zo!
=> Labelen om duidelijk te maken.

vb. Relatietype OVERZIET:
- Rol van overste labelen als 1
- Rol van ondergeschikte labelen als 2



Relaties in ER Model

Beperkingen op relatietypes: Mogelijke combinaties entiteiten

1. **Cardinaliteitsratio** (*connectiviteit labelen*)

Zegt iets over Kunnen!

vb. BEHEERT => DEPARTEMENT:WERKNEMER is **1:1**
vb. WERKT_VOOR => DEPARTEMENT:WERKNEMER is **1:N**
vb. WERKT_OP => PROJECT:WERKNEMER is **M:N**

2. **Deelnamebeperking**: Totaal (dubbele lijn) of Partiëel (enkele lijn)

Zegt iets over Moeten!

vb. Elke WERKNEMER werkt voor een DEPARTEMENT en
elk DEPARTEMENT heeft een WERKNEMER dat ervoor werkt => **T-T**
vb. Niet elke WERKNEMER beheert een DEPARTEMENT maar
elk DEPARTEMENT wordt beheerd door een WERKNEMER => **P-T**
vb. Niet elke WERKNEMER heeft een ondergeschikte of overste => **P-P**

Voor binaire relatie 3 mogelijkheden per aspect, ternair meer mogelijkheden.

Relaties in ER Model

Relatietypes kunnen attributen hebben.

Vb. WERKNEMER beheert een DEPARTEMENT vanaf een bepaalde Startdatum

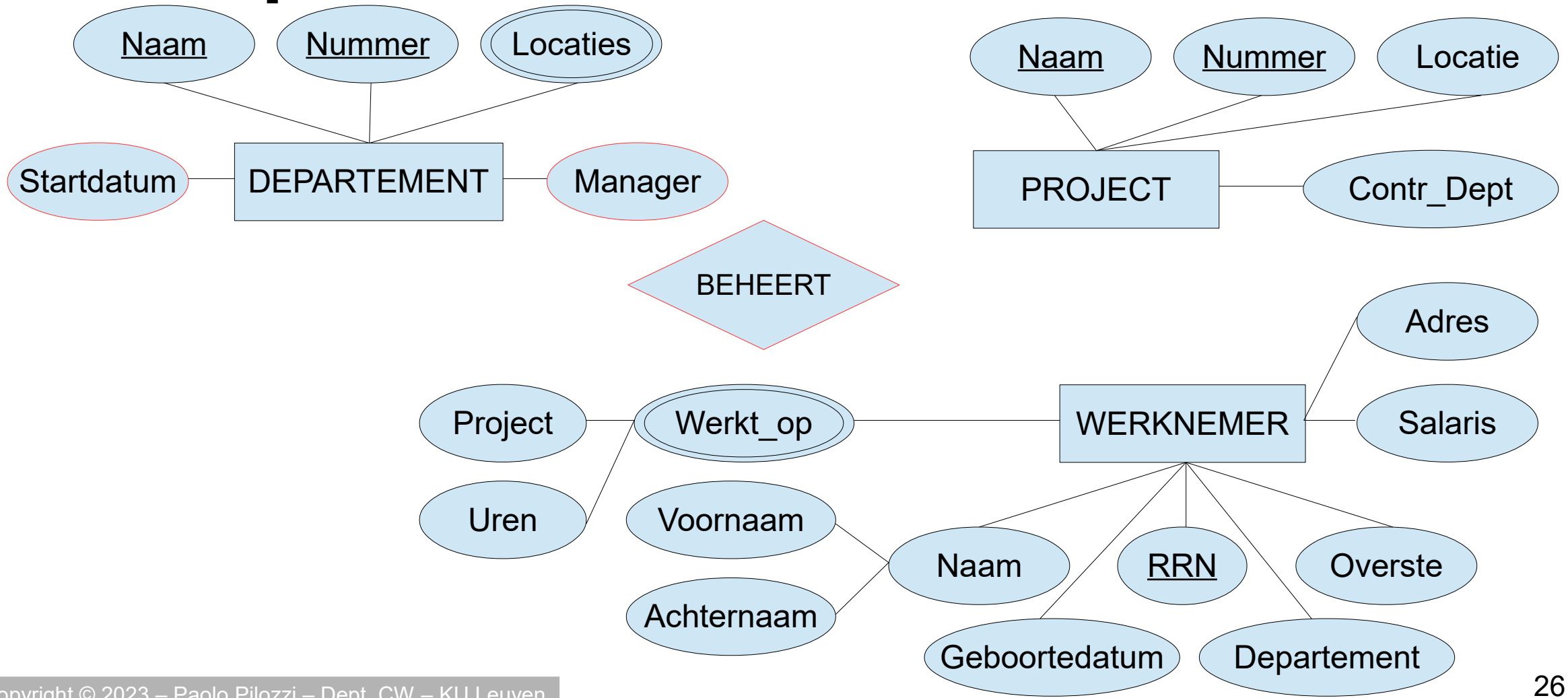
Vb. WERKNEMER werkt een #Uren per week op een PROJECT

De mogelijkheden:

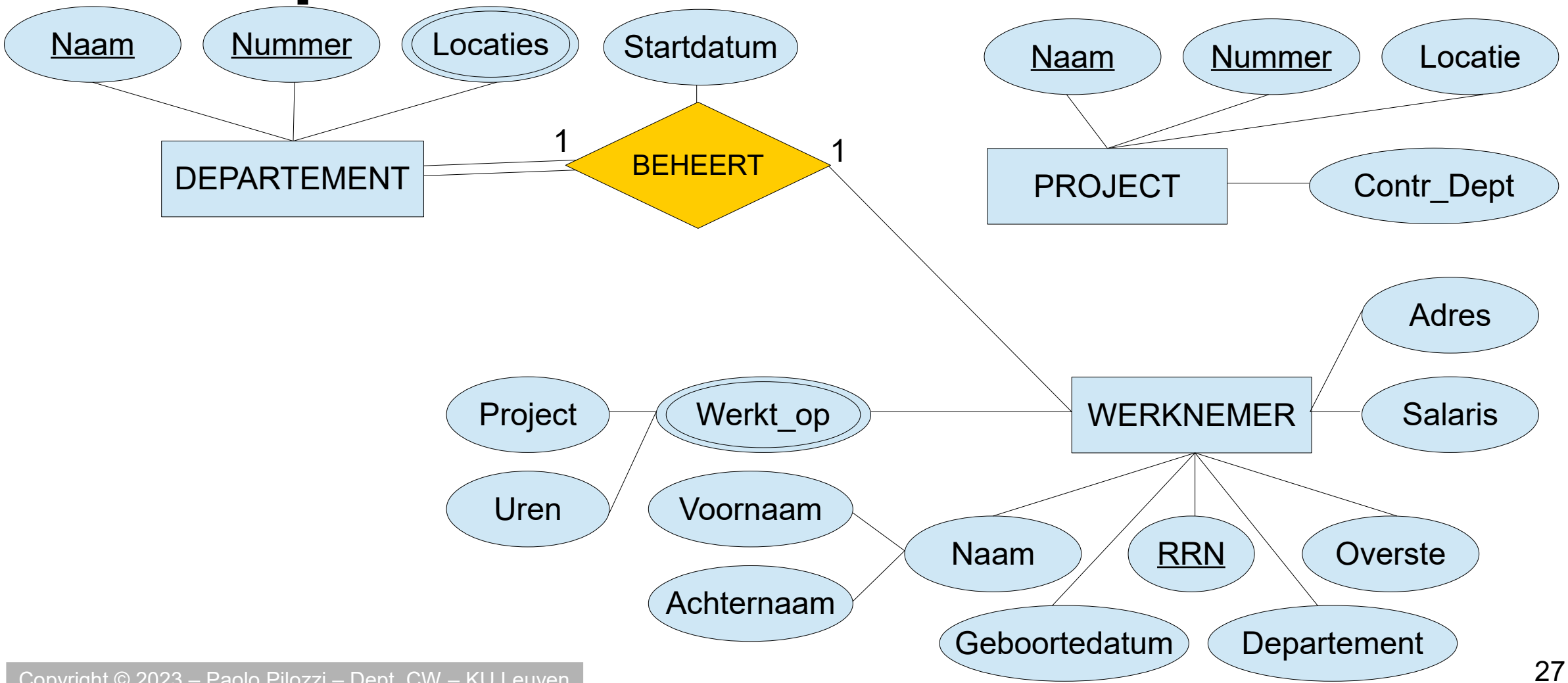
* Bij een 1:1 relatietype kan attribuut bij relatietype of bij één der entiteitstypes.
  vb. Bij BEHEERT kan Startdatum bij relatietype, WERKNEMER of DEPARTEMENT
* Bij een 1:N relatietype kan attribuut bij relatietype of bij entiteitstype a.d. N kant.
  vb. Moest WERKT_VOOR een Startdatum hebben dan kan dit enkel bij het relatietype of de WERKNEMER, maar niet bij DEPARTEMENT
* Bij een M:N relatie moet attribuut by relatietype.
  vb. Bij WERKT_OP moet #Uren bij het relatietype

De keuze hierin is (in principe) vrij. MAAR hou de dingen duidelijk! => Best bij relatietype.
Functionele behoeften kunnen de keuze beïnvloeden: vb. #opzoekingen beperken

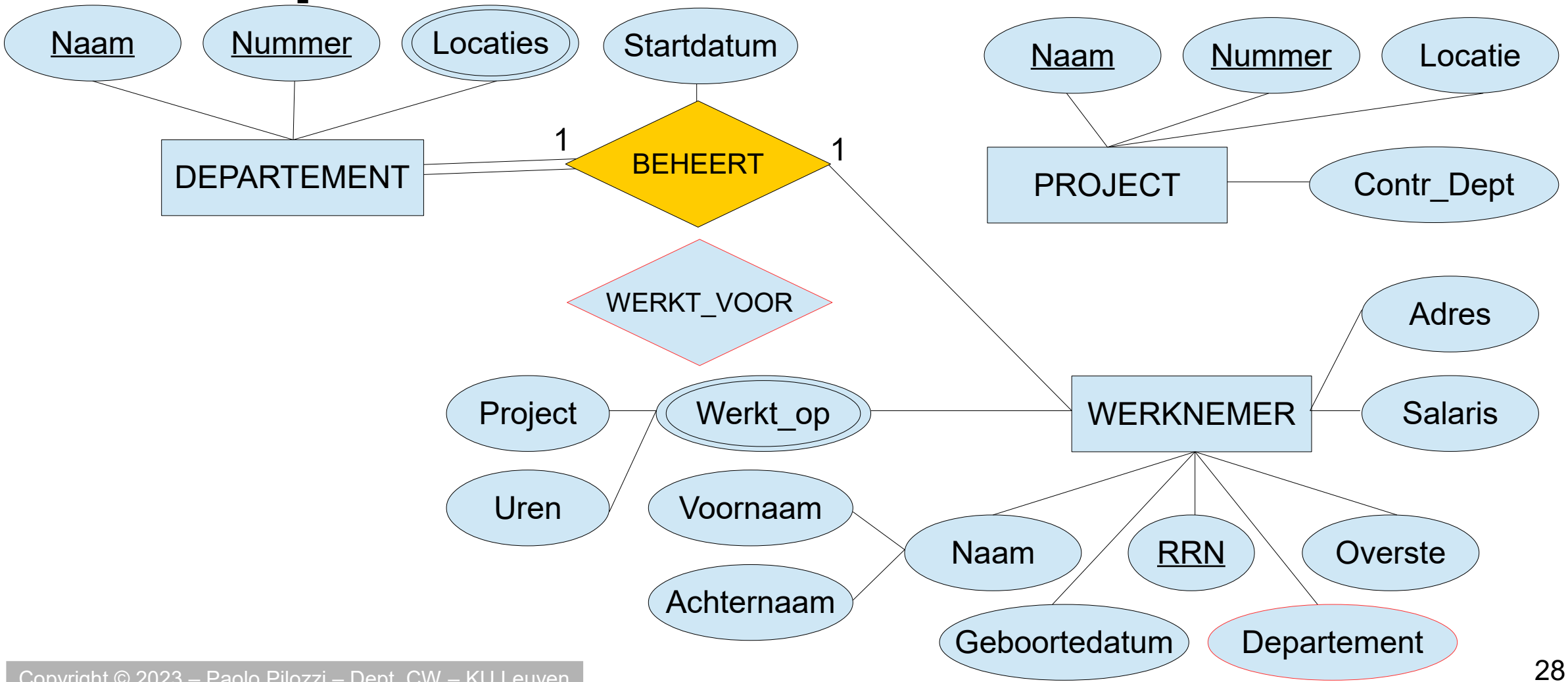
Doorlopend Voorbeeld: BEDRIJF



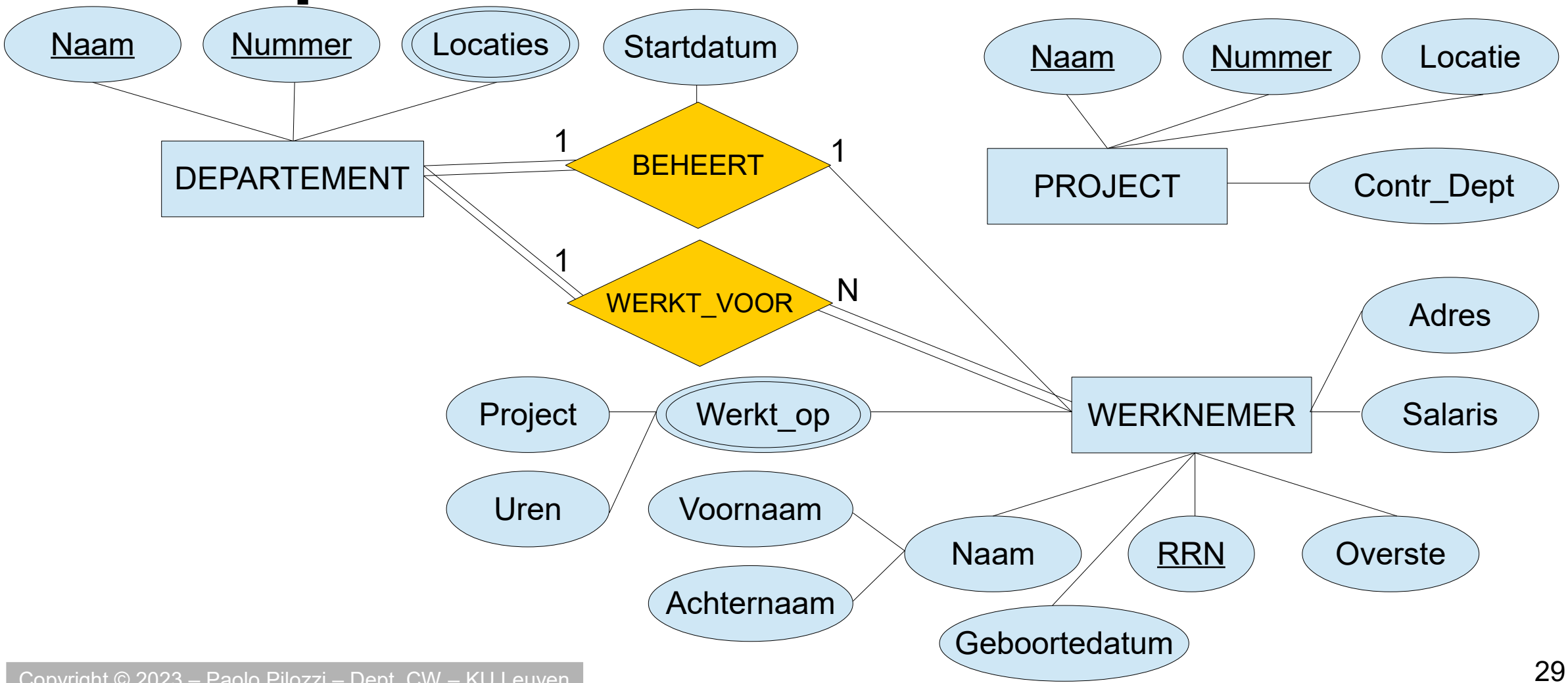
Doorlopend Voorbeeld: BEDRIJF



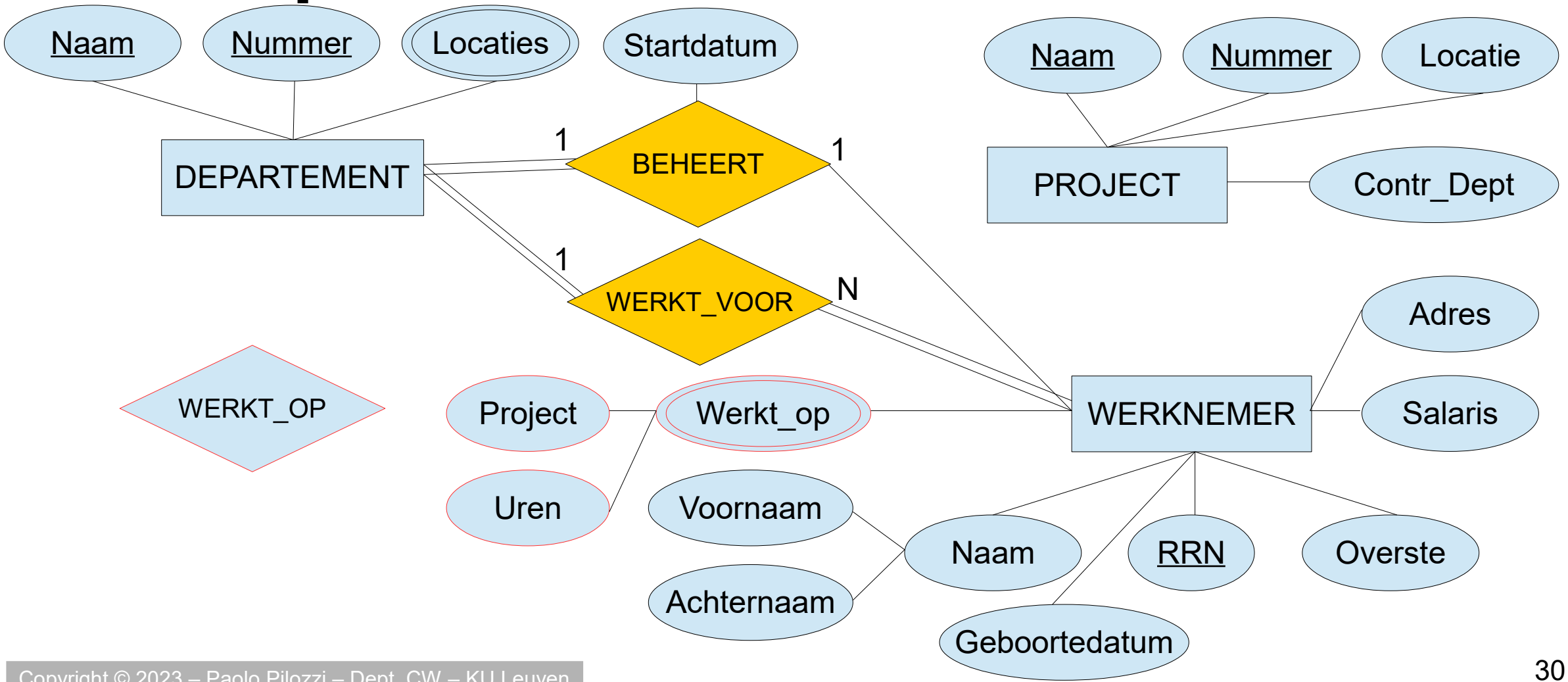
Doorlopend Voorbeeld: BEDRIJF



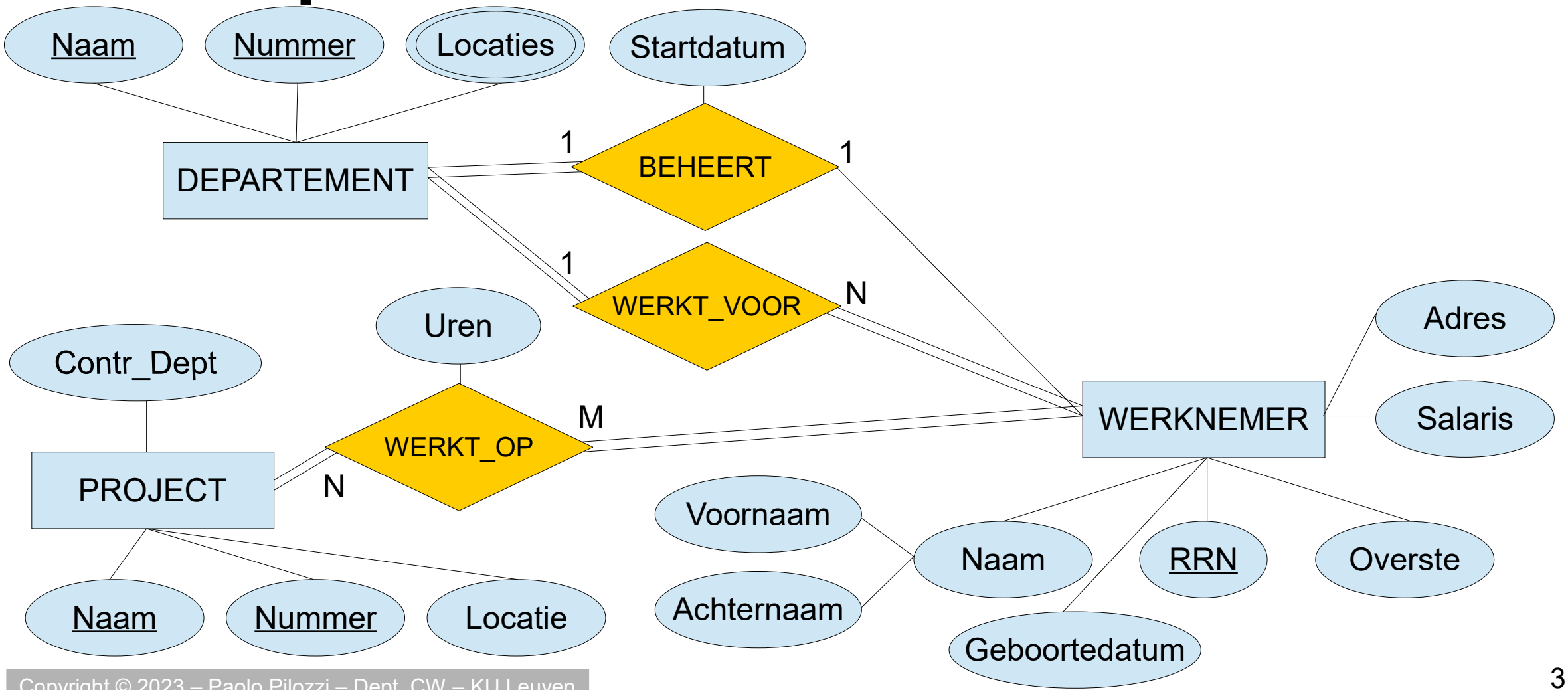
Doorlopend Voorbeeld: BEDRIJF



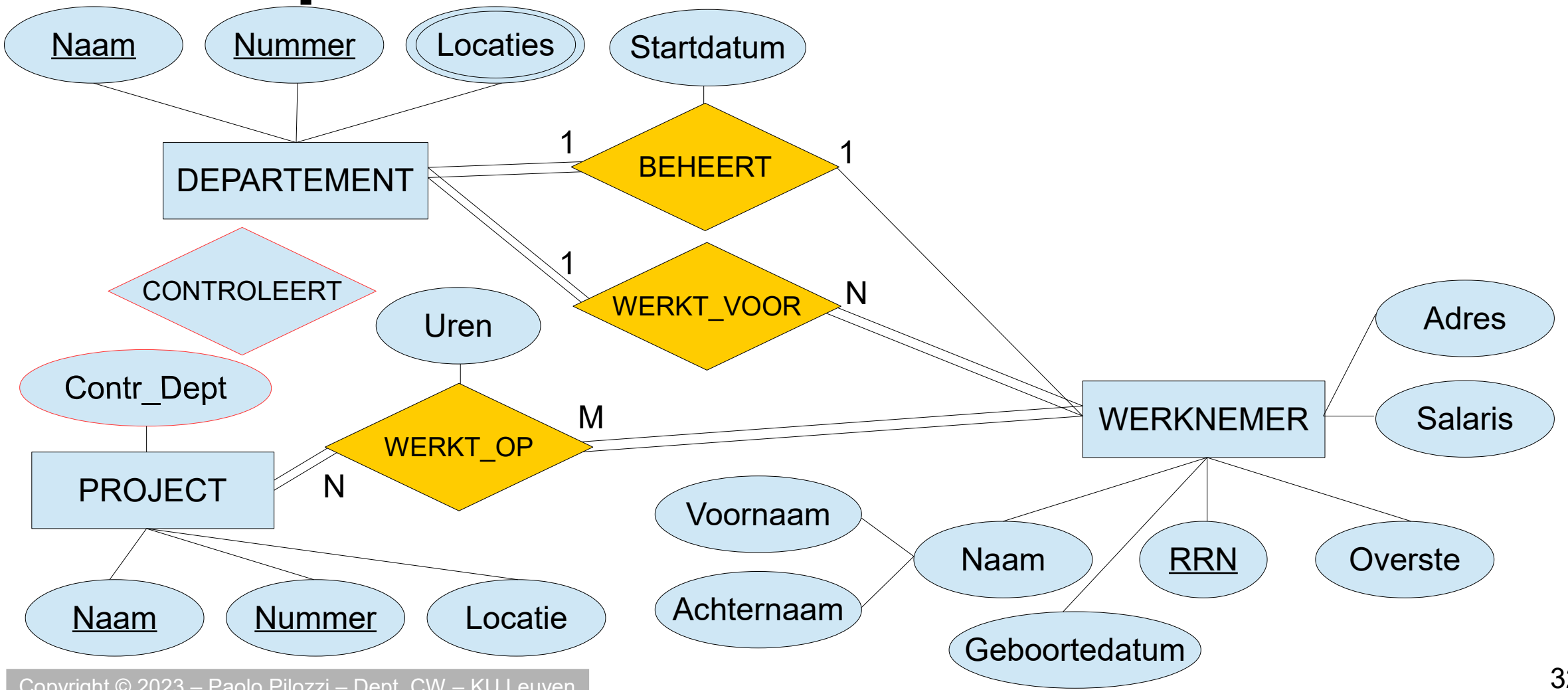
Doorlopend Voorbeeld: BEDRIJF



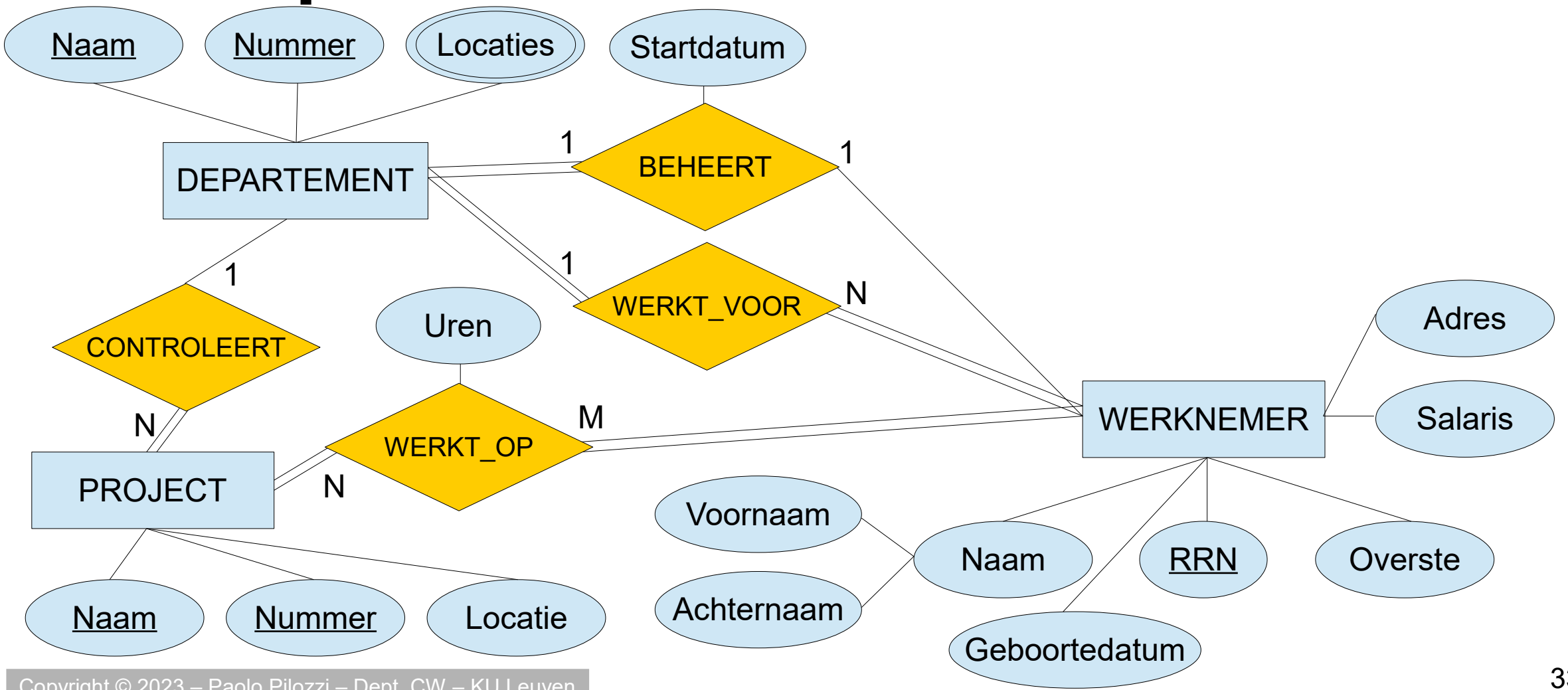
Doorlopend Voorbeeld: BEDRIJF



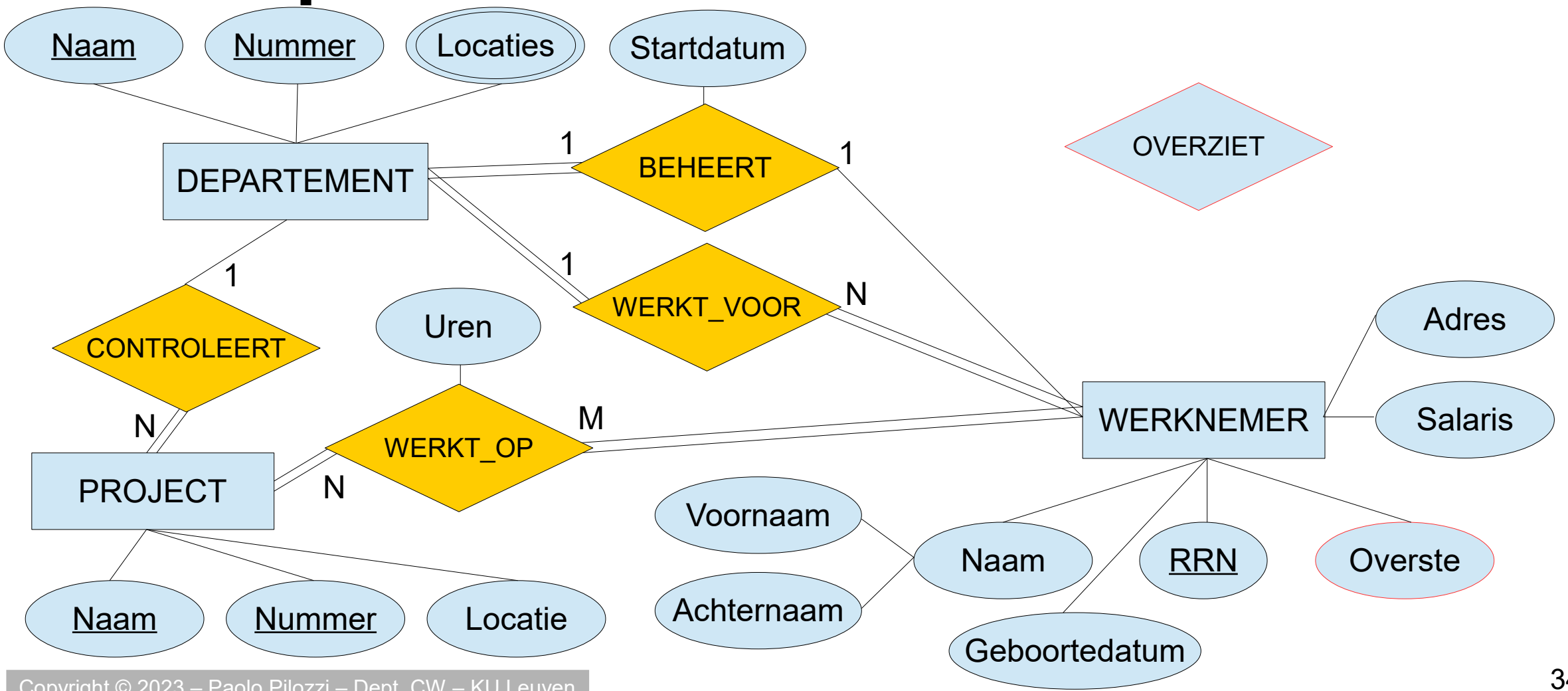
Doorlopend Voorbeeld: BEDRIJF



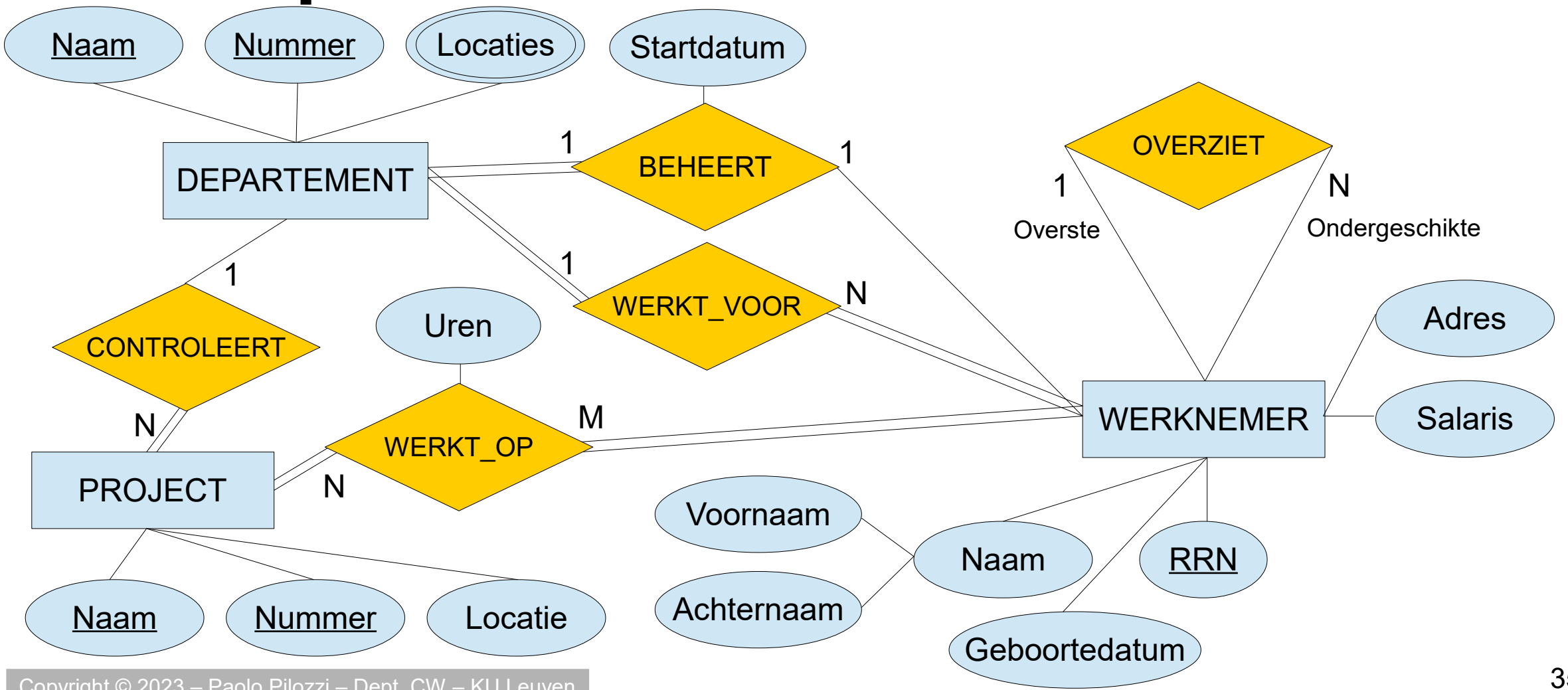
Doorlopend Voorbeeld: BEDRIJF



Doorlopend Voorbeeld: BEDRIJF



Doorlopend Voorbeeld: BEDRIJF



Zwakke Entiteitstypes (vs. Sterke/Regulier)

= Entiteitstype zonder eigen sleutel (Enkel als het moet!).

In ER Diagramma: rechthoek met dubbele omlijning

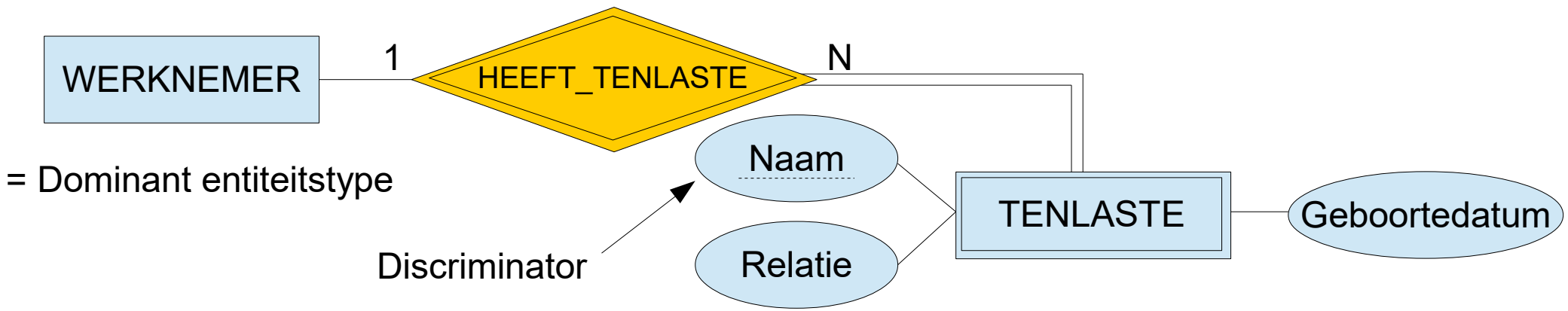
=> Entiteit bepaald via ander entiteitstype (*bestaansafhankelijk*): enkel via relatietype

In ER Diagramma: ruit met dubbele omlijning

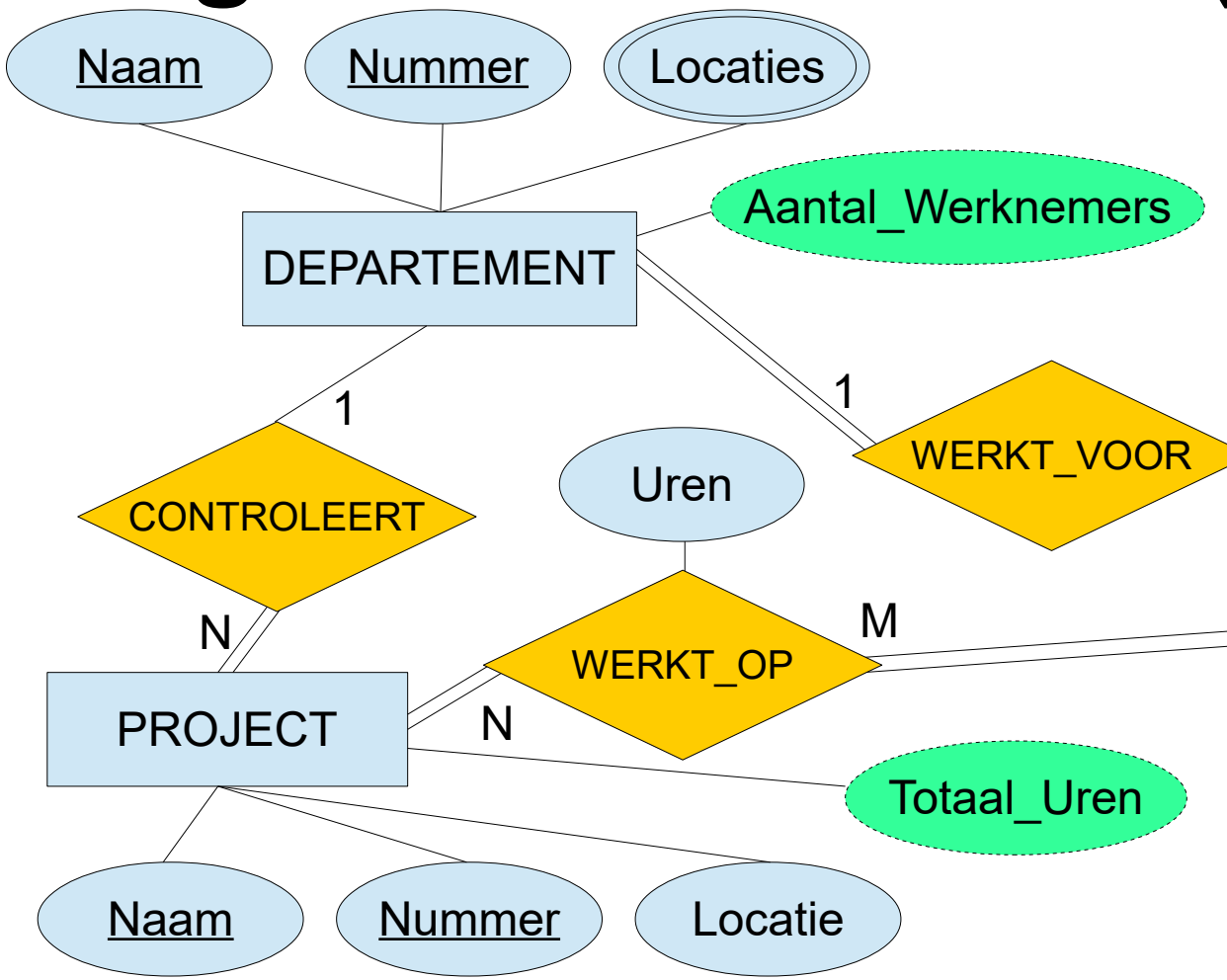
=> Vereist een **totale deelname**: elke zwakke entiteit extact 1 **eigenaar**

=> **Partiëel sleutelattribuut** (*identificatieafhankelijk*): identificatie per eigenaar

In ER Diagramma: gestippeld onderstrepen



Afgeleide Attributen (vb. aggregaten)



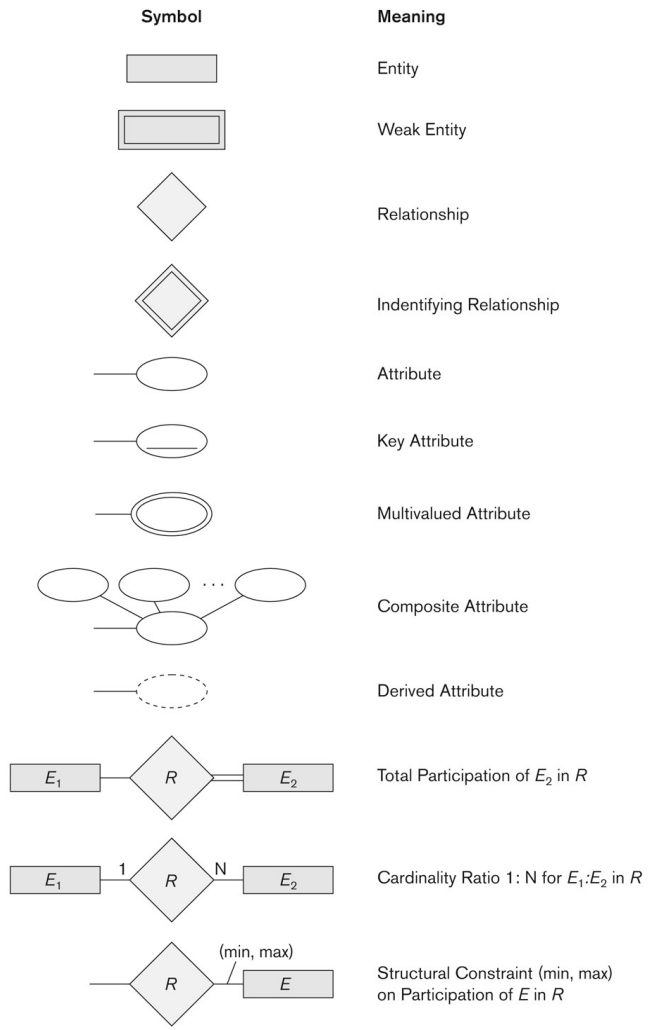
Vb. Werknemers per Departement via WERKT_VOOR

Vb. Totaal aantal uren gepresteerd op project via WERKT_OP

Conventies Notatie & ER Diagramma

* Entiteitstype
  - Enkelvoud
  - Hoofdletters
  - Geen spaties (gebruik underscore '_')
    - Vb. WERKNEMER
* Relatietype
  - Vervoegd werkwoord
  - Hoofdletters
  - Geen spaties
    - Vb. WERKT_VOOR
* Attributen en Rollen
  - Start met hoofdletter (en ook na underscore)
    - vb. Geb_Datum

Figure 3.14
Summary of the notation for ER diagrams.



Min-max Annotatie (vs. Cardinatliteitsratio's)

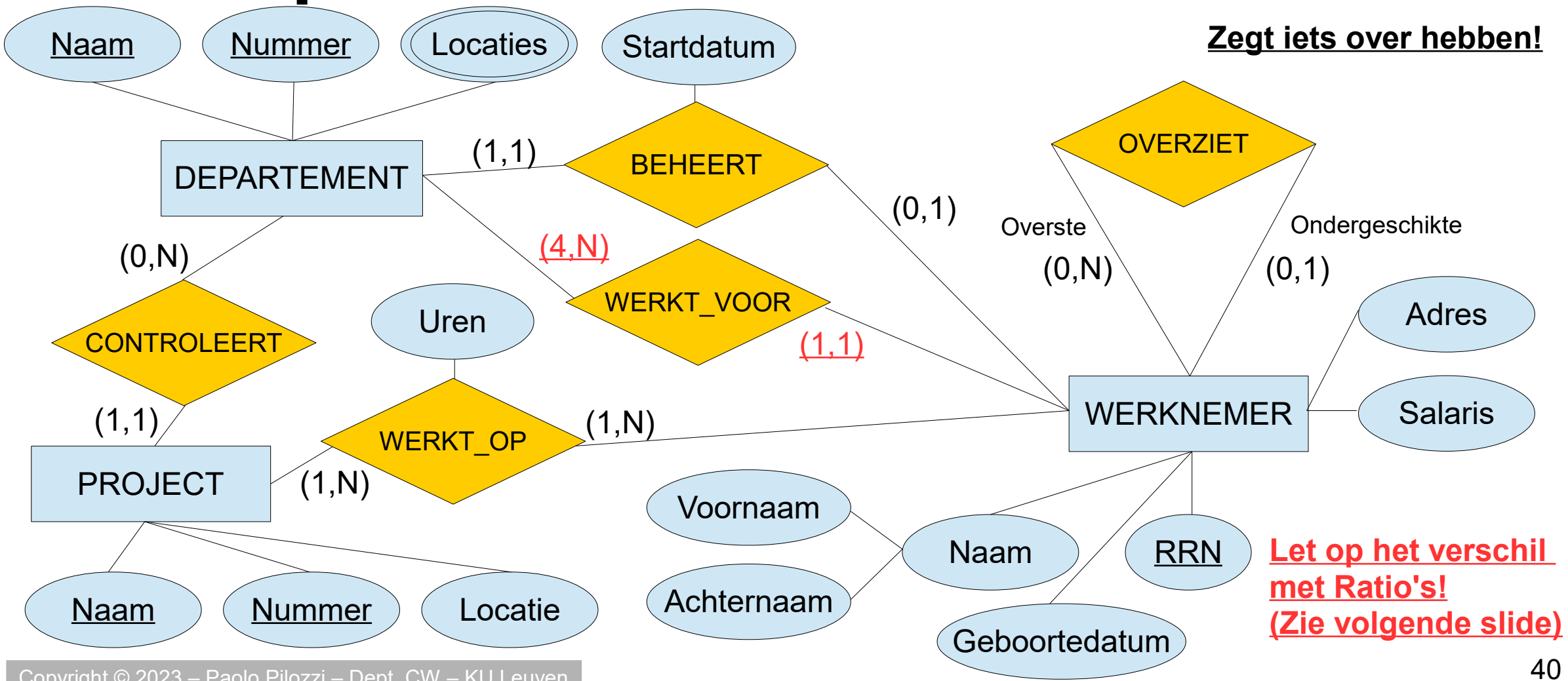
* Min-max annotatie voor **beperkingen op relatietypes** (= exacter)
  Wordt geschreven bij elke participatie van een entiteitstype in een relatietype als (min,max) i.p.v. 1, N, M zoals eerder.

* **Betekenis**: Elke entiteit komt minstens **min** keer en maximaal **max** keer voor in relaties van het relatietype, waarbij **0 ≤ min ≤ max** en **max ≥ 1**
* **Gevolg**: als **min = 0** dan partiële participatie anders (**min > 0**) is het totale

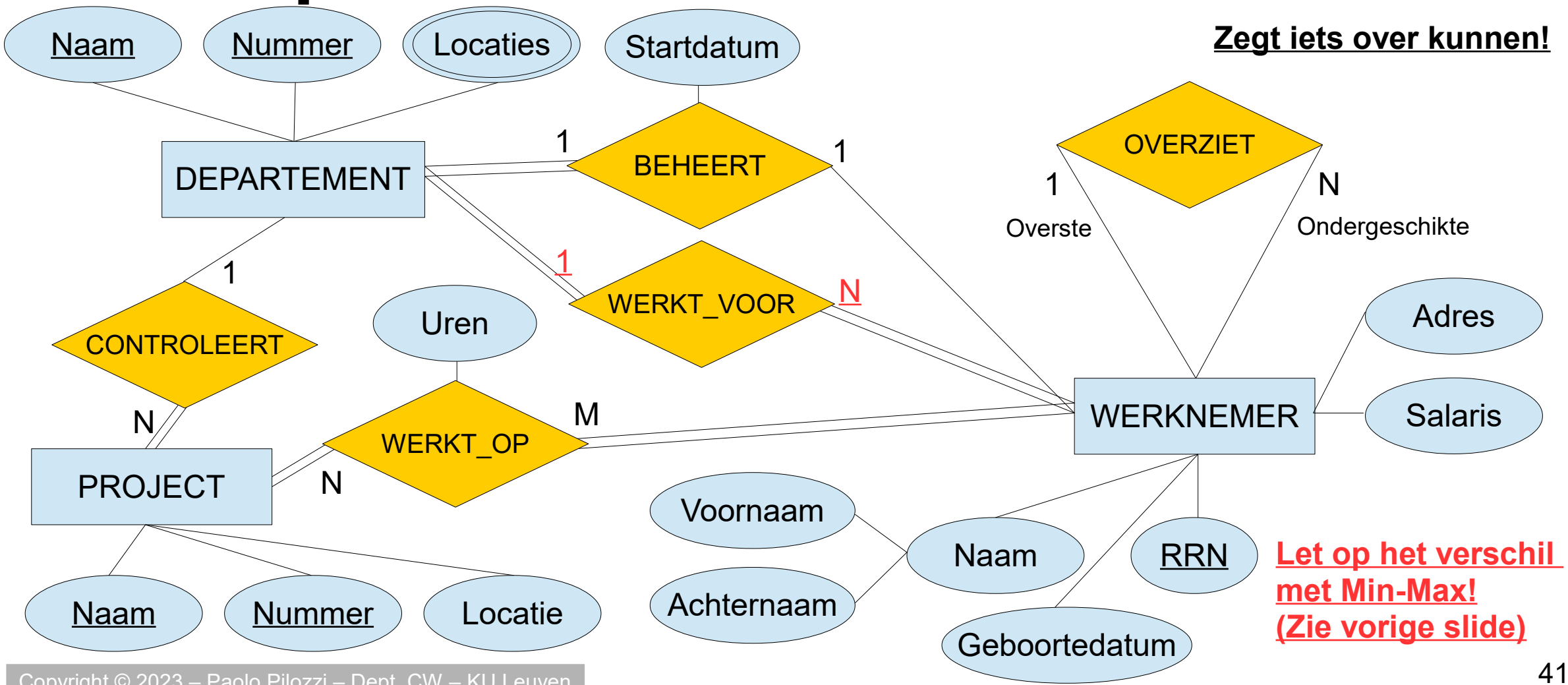
** Min-Max zegt iets over hebben!*

In ER Diagramma steeds consistent zijn! Of Ratio's of Min-Max!
Dubbele verbindingslijnen tussen entiteitstypes en relatietypes niet nodig!

Doorlopend Voorbeeld: BEDRIJF



Doorlopend Voorbeeld: BEDRIJF



2.2 ERM → Relationeel Schema

Stappen ER Model omzetten naar een Relationeel Model (RM)
Hoofdstuk 7.1 – Oefenzitting 1

1. Voor elk regulier entiteitstype
2. Voor elk zwak entiteitstype
3. Voor elk binair 1:1 relatietype
4. Voor elk binair 1:N relatietype
5. Voor elk binair M:N relatietype
6. Voor elk meerwaardig attribuut
7. Voor elk relatietype van graad > 2

Opgelet! In RM betekent een relatie niet hetzelfde als in ER model!!!!

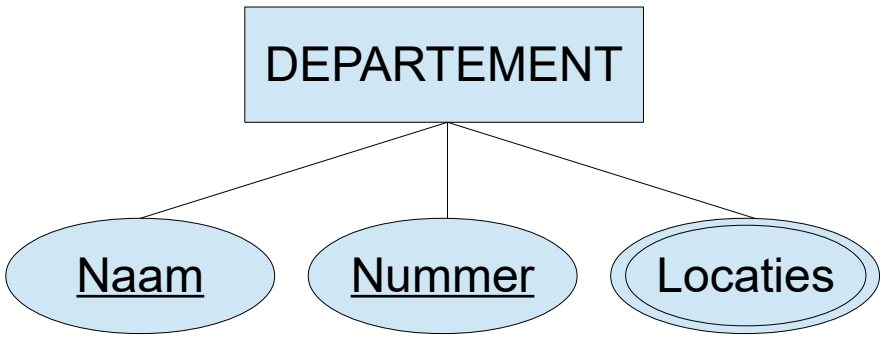
Opmerkingen

* Entiteitstypes en relatietypes worden beide
  relaties (tabellen) in een Relationeel Model Schema (RMS of RS)
* ER Model is conceptueel terwijl RM Schema implementatief is
* Entiteitstypes: sterk/regulier of zwak
* Relatietypes: binair of hogere-graad
* Attributen: enkelvoudig, samengesteld, meerwaardig of complex
  - Attributen hebben domeinen bepaald door Attribuuttypes
* Beperkingen op verbanden (relatietypes) en attributen

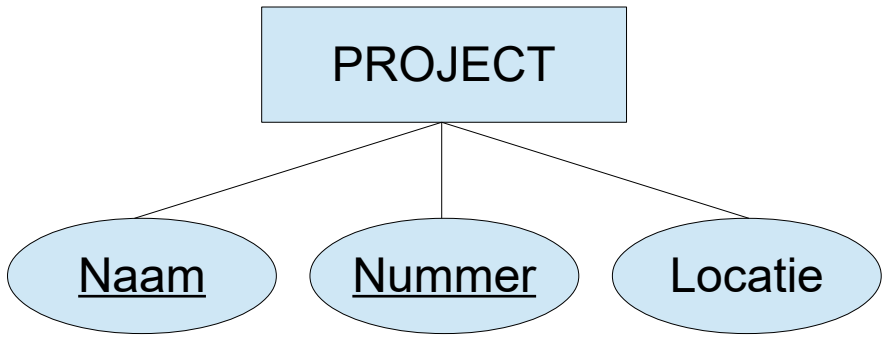
Stap 1 – Elk regulier entiteitstype

Voor elk regulier/sterk entiteitstype E in ERM

* Maak een relatie R in RMS
* Met alle enkelvoudige attributen van E
* Samengestelde attributen: enkelvoudige attribuutonderdelen opdelen
* Meerwaardige attributen: zie stap 6
* Kies een primaire sleutel op basis van sleutelattributen in ERM



DEPARTEMENT: Dnaam, Dnr



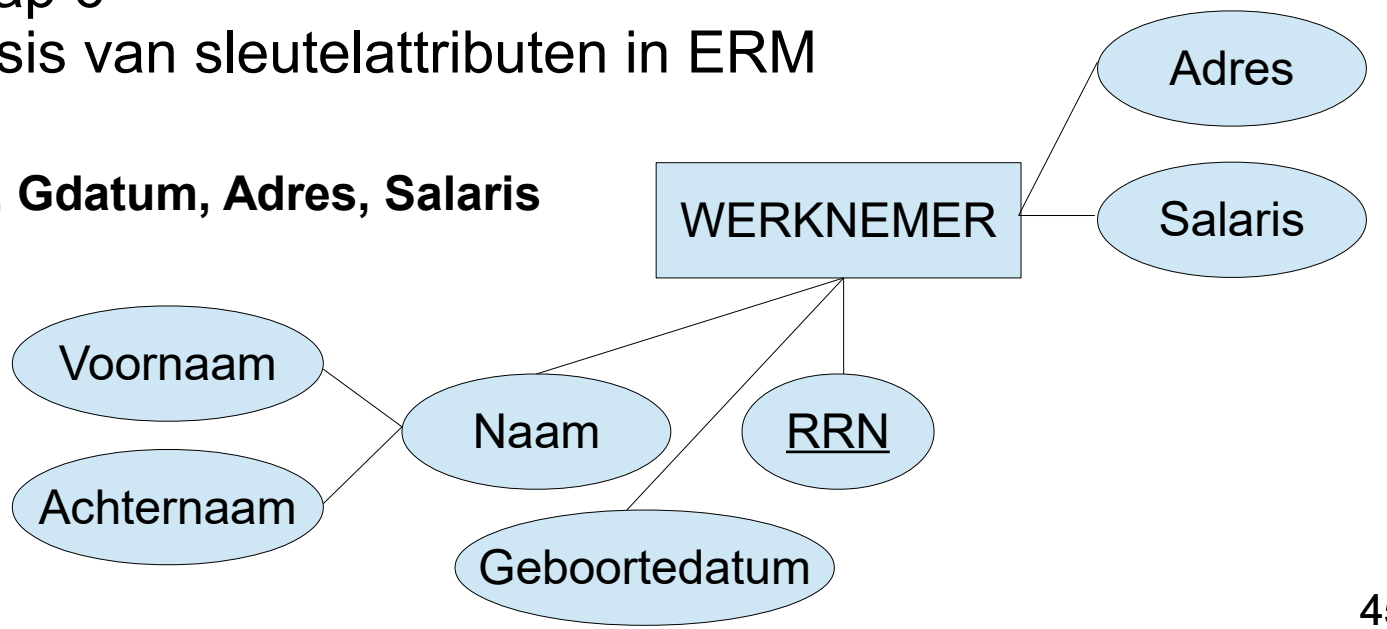
PROJECT: Pnaam, Pnr, Ploc

Stap 1 – Elk regulier entiteitstype

Voor elk regulier/sterk entiteitstype E in ERM

* Maak een relatie R in RMS
* Met alle enkelvoudige attributen van E
* Samengestelde attributen: enkel de enkelvoudige attribuutonderdelen
* Meerwaardige attributen: zie stap 6
* Kies een primaire sleutel op basis van sleutelattributen in ERM

WERKNEMER: Vnaam, Anaam, <u>Rrn</u>, Gdatum, Adres, Salaris



Doorlopend Voorbeeld: BEDRIJF

WERKNEMER

Vnaam	Anaam	Rrn	Gdatum	Adres	Salaris

DEPARTEMENT

Dnaam	Dnr

PROJECT

Pnaam	Pnr	Ploc

Stap 2 – Elk zwak entiteitstype

Voor elk zwak entiteitstype E in ERM

* Maak een relatie R in RMS
* Met alle enkelvoudige attributen van E
* Samengestelde attributen: enkelvoudige attribuutonderdelen opdelen
* Meerwaardige attributen: zie stap 6
* Verwijssleutel is de primaire sleutel van eigenaar
* Kies primaire sleutel op basis van verwijssleutel en discriminator

TENLASTE: <u>Errn</u>, <u>Naam</u>, Gdatum, Relatie

Doorlopend Voorbeeld: BEDRIJF

WERKNEMER

Vnaam	Anaam	Rrn	Gdatum	Adres	Salaris

DEPARTEMENT

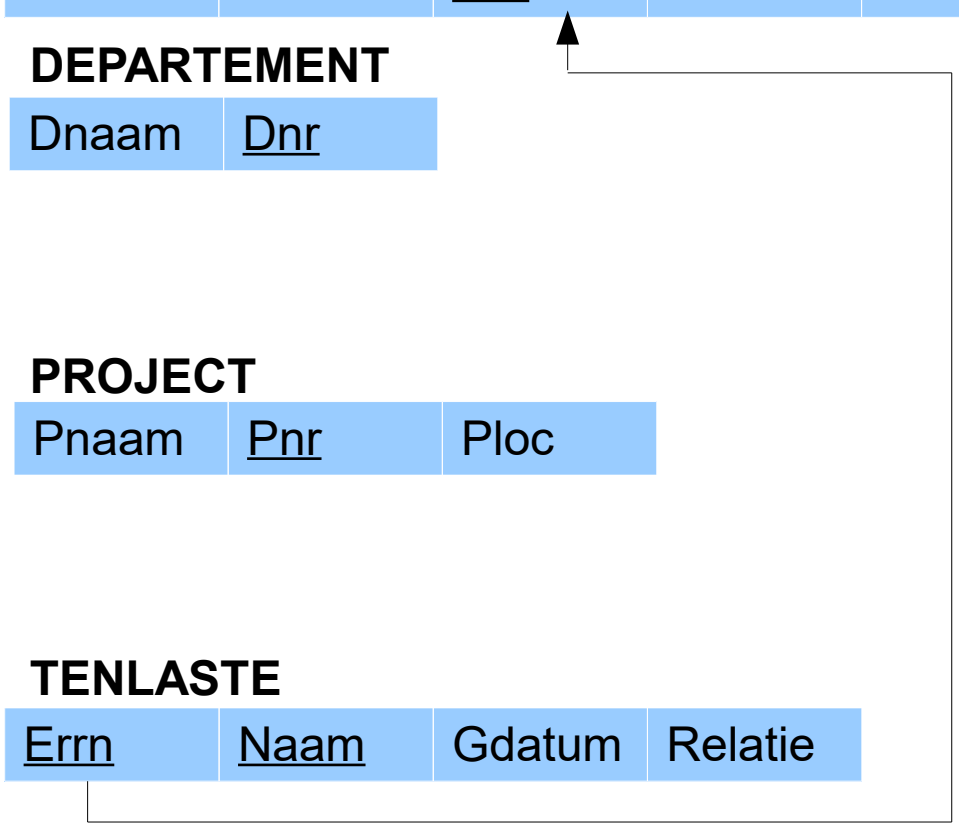
Dnaam	Dnr

PROJECT

Pnaam	Pnr	Ploc

TENLASTE

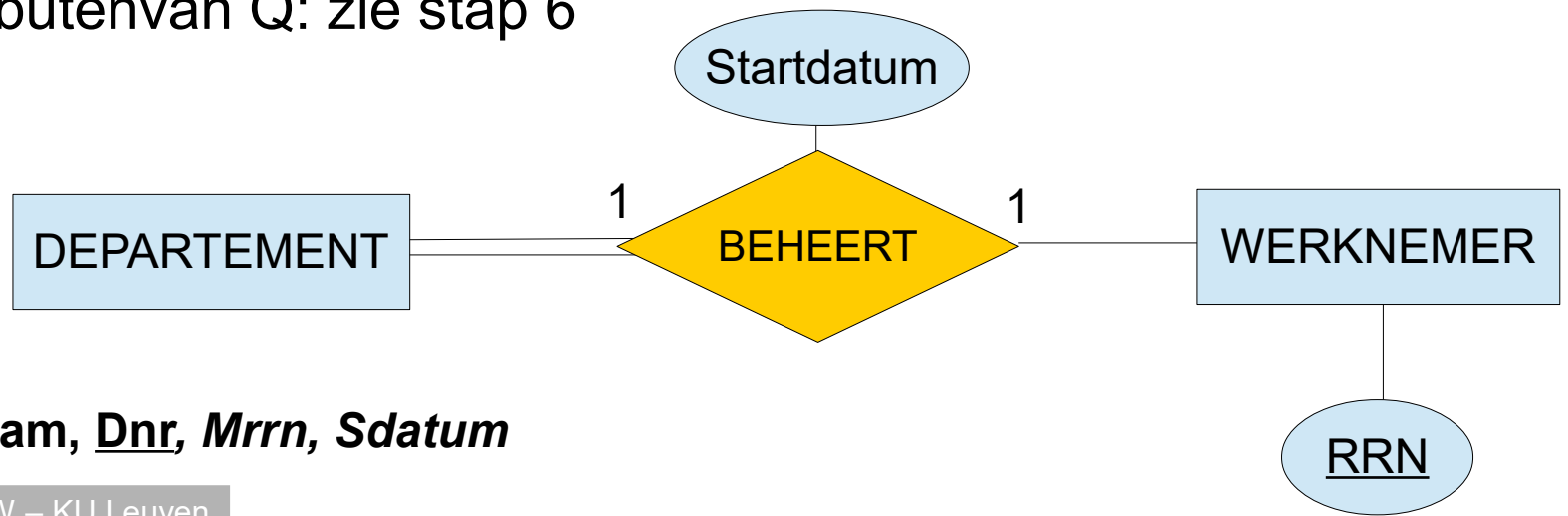
Errn	Naam	Gdatum	Relatie



Stap 3 – Elk binair 1:1 relatietype

Voor elk binair 1:1 relatietype Q in ERM

* Er dienen relaties S en T voor deelnemende entiteitstypes te zijn in Q
* Neem primaire sleutel van S op in T of omgekeerd
* Vermijdt NULL waarden (kies daarom entiteitstype met totale participatie)
* Neem ook enkelvoudige attributen van Q op
* Samengestelde attributen van Q: enkelvoudige attribuutonderdelen opdelen
* Meerwaardige attributenvan Q: zie stap 6



DEPARTEMENT: Dnaam, <u>Dnr</u>, *Mrrn, Sdatum*

Doorlopend Voorbeeld: BEDRIJF

WERKNEMER

Vnaam	Anaam	Rrn	Gdatum	Adres	Salaris

DEPARTEMENT

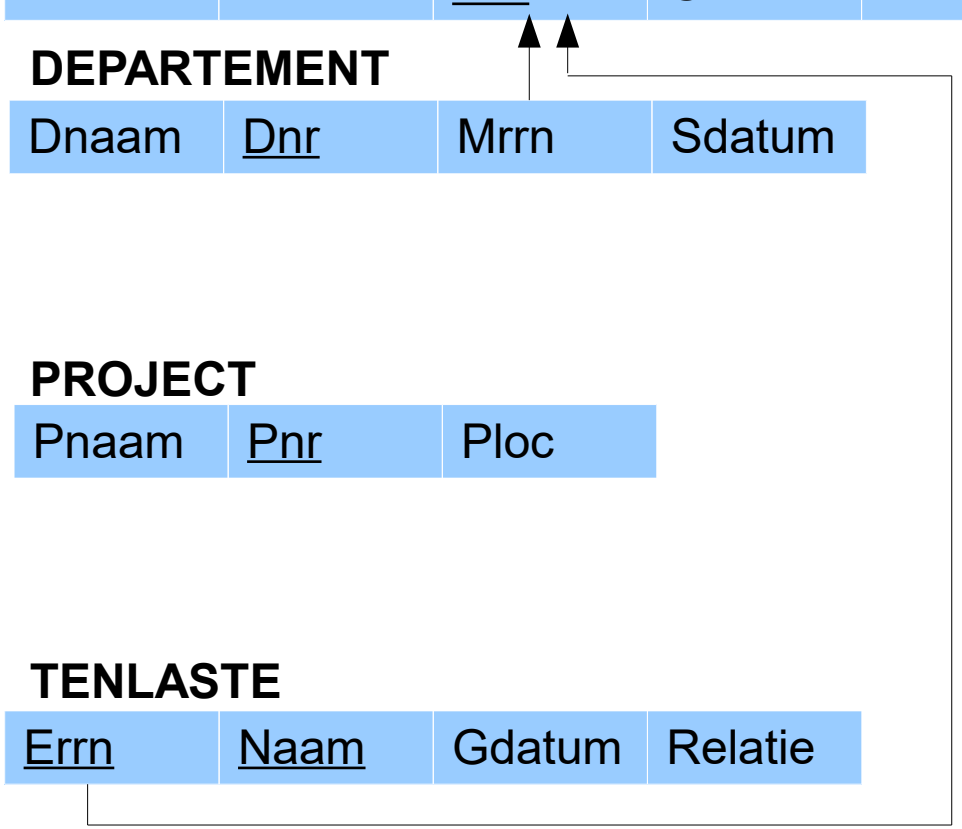
Dnaam	Dnr	Mrrn	Sdatum

PROJECT

Pnaam	Pnr	Ploc

TENLASTE

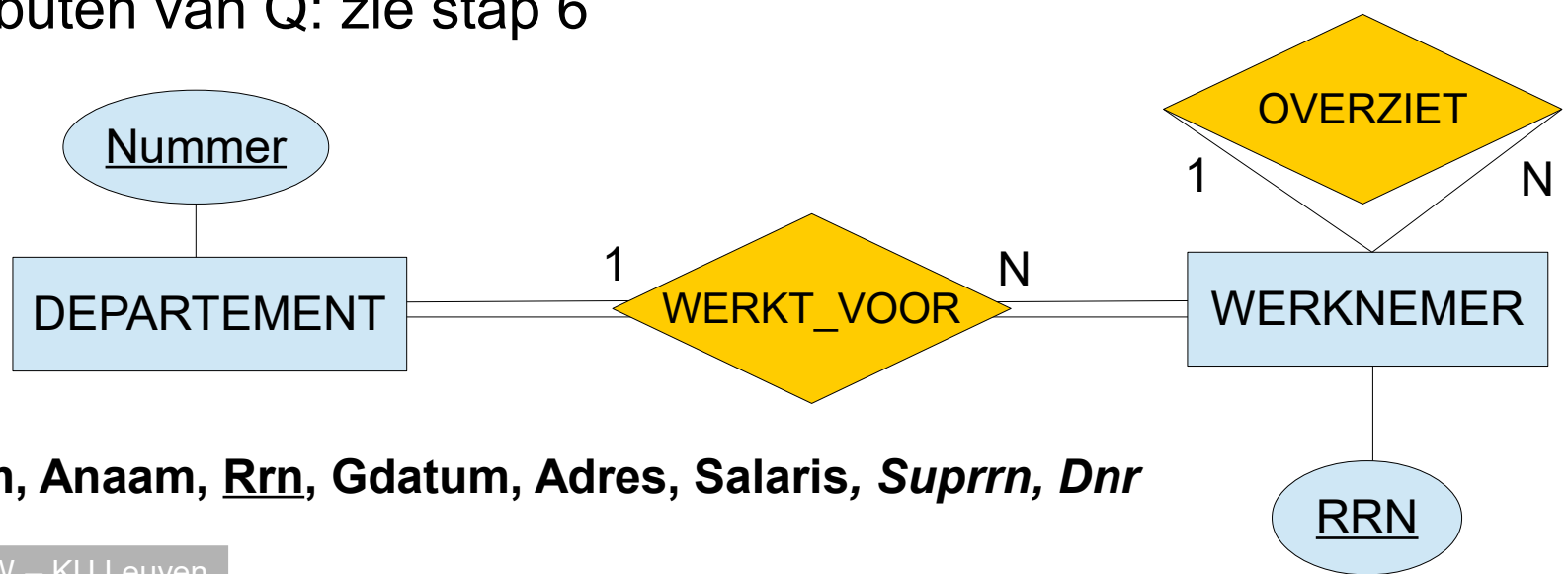
Errn	Naam	Gdatum	Relatie



Stap 4 – Elk binair 1:N relatietype

Voor elk binair 1:N relatietype Q in ERM

* Er dienen relaties S en T voor deelnemende entiteitstypes te zijn in Q
* Neem primaire sleutel van S op in T (N kant)
* Neem ook enkelvoudige attributen van Q op
* Samengestelde attributen van Q: enkelvoudige attribuutonderdelen opdelen
* Meerwaardige attributen van Q: zie stap 6



WERKNEMER: Vnaam, Anaam, Rrn, Gdatum, Adres, Salaris, *Suprrn, Dnr*

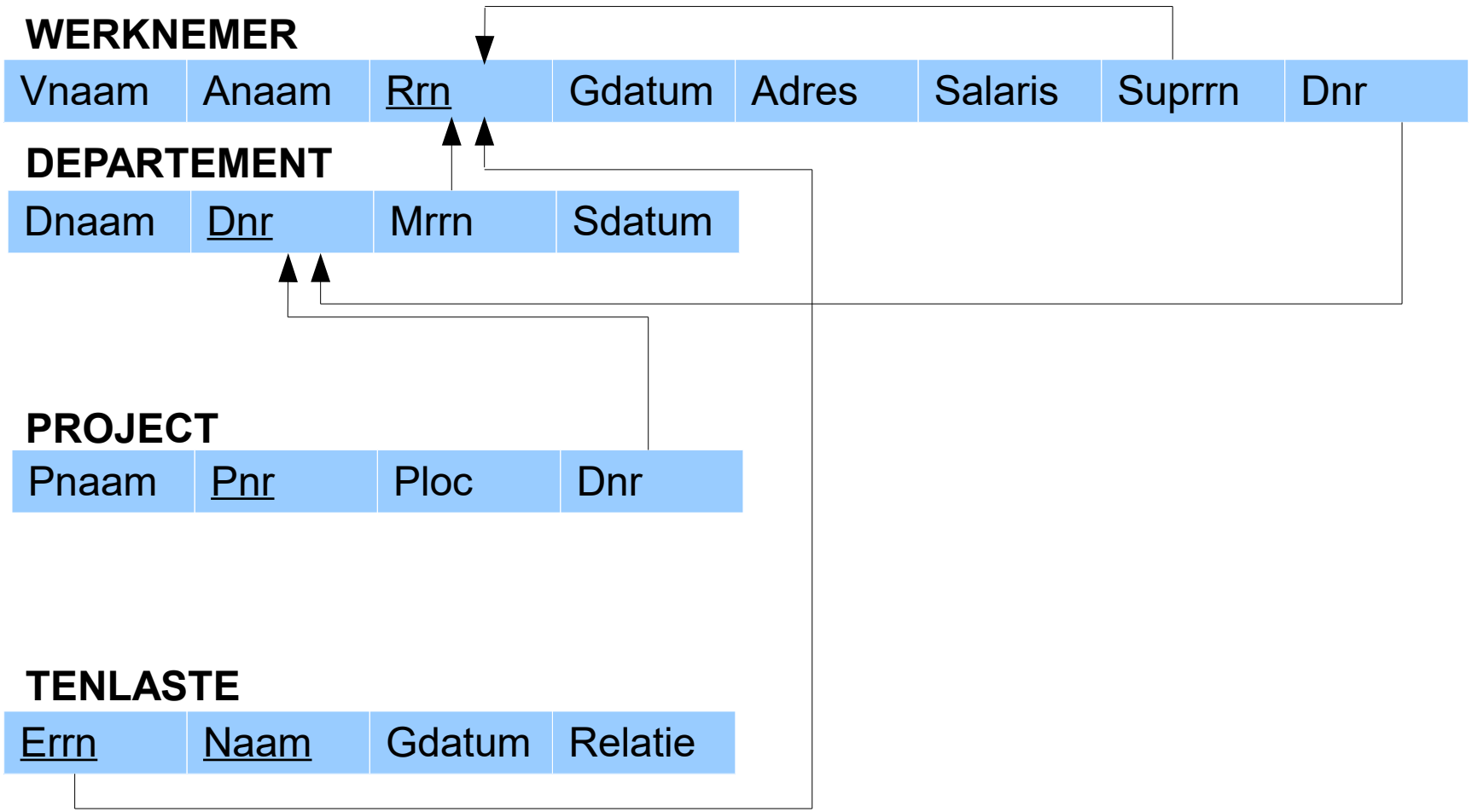
Stap 4 – Elk binair 1:N relatietype

Voor elk binair 1:N relatietype Q in ERM

* Er dienen relaties S en T voor deelnemende entiteitstypes te zijn in Q
* Neem primaire sleutel van S op in T (N kant)
* Neem ook enkelvoudige attributen van Q op
* Samengestelde attributen van Q: enkelvoudige attribuutonderdelen opdelen
* Meerwaardige attributen van Q: zie stap 6

PROJECT: Pnaam, <u>Pnr</u>, Ploc, *Dnr*

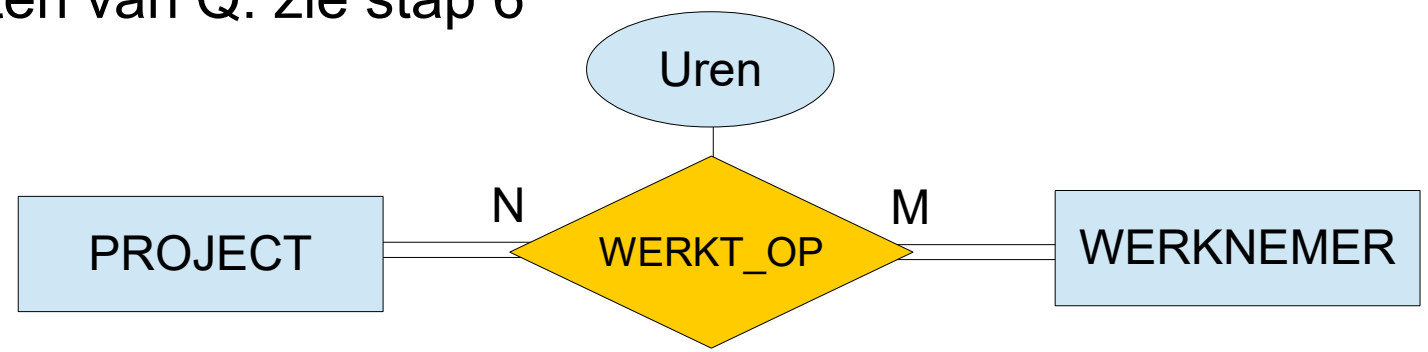
Doorlopend Voorbeeld: BEDRIJF



Stap 5 – Elk binair M:N relatietype

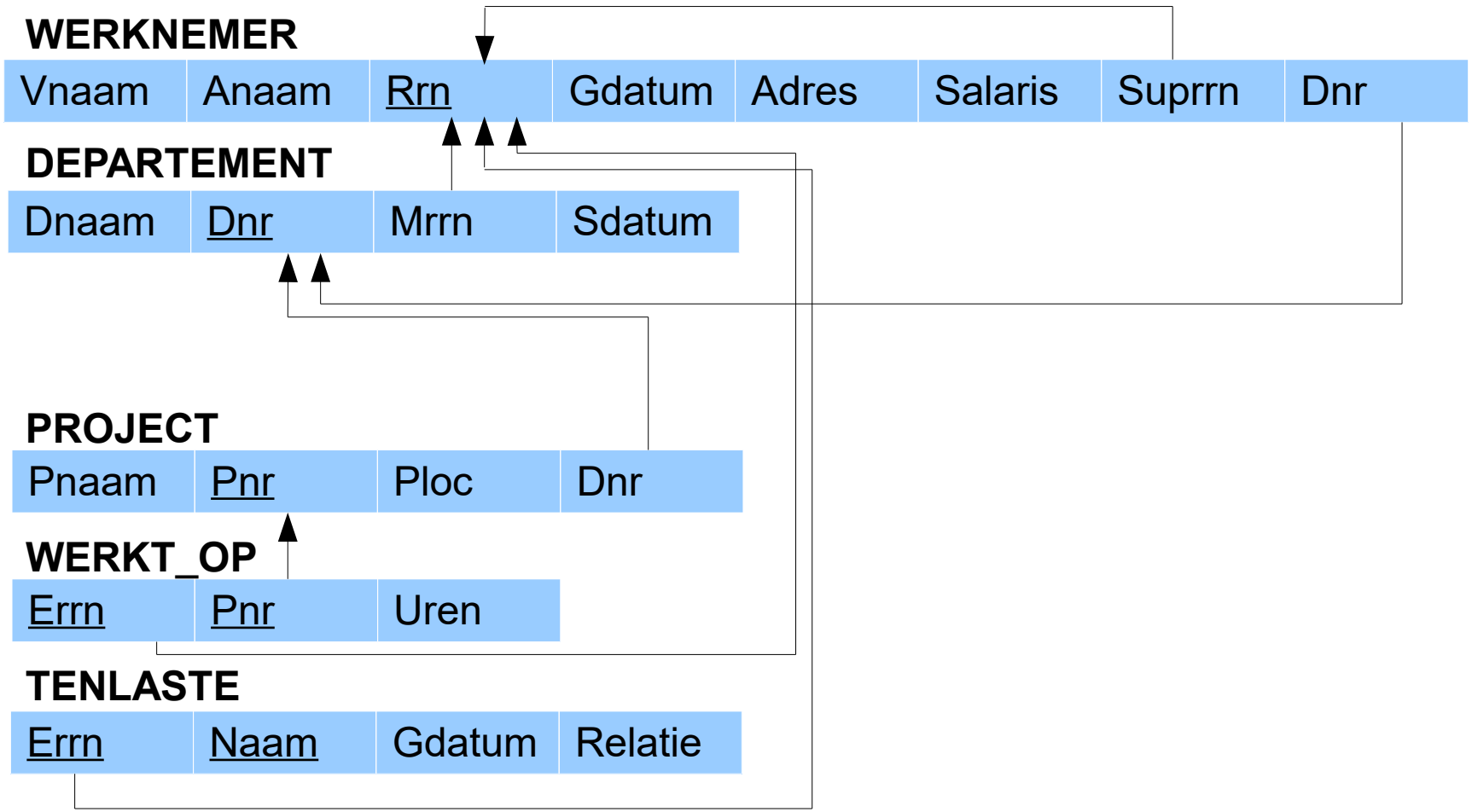
Voor elk binair M:N relatietype Q in ERM

* Maak nieuwe relatie S in schema
* Neem primaire sleutels van deelnemende entiteitstypes in Q op
* Verwijssleutels samen vormen primaire sleutel van S
* Neem ook enkelvoudige attributen van Q op
* Samengestelde attributen van Q: enkelvoudige attribuutonderdelen opdelen
* Meerwaardige attributen van Q: zie stap 6



WERKT_OP: <u>Errn</u>, <u>Pnr</u>, Uren

Doorlopend Voorbeeld: BEDRIJF

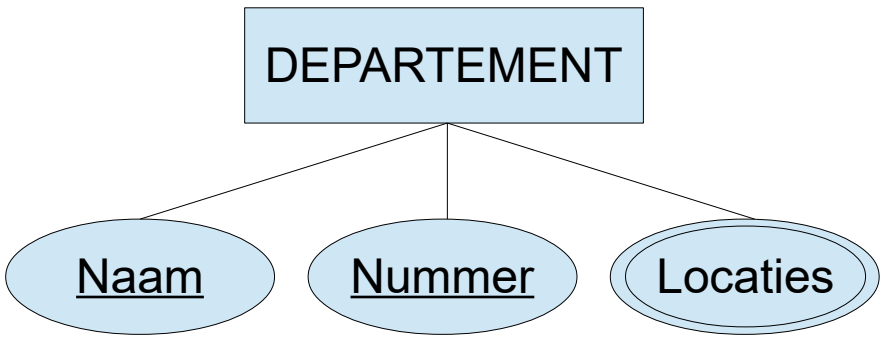


Stap 6 – Elk meerwaardig attribuut

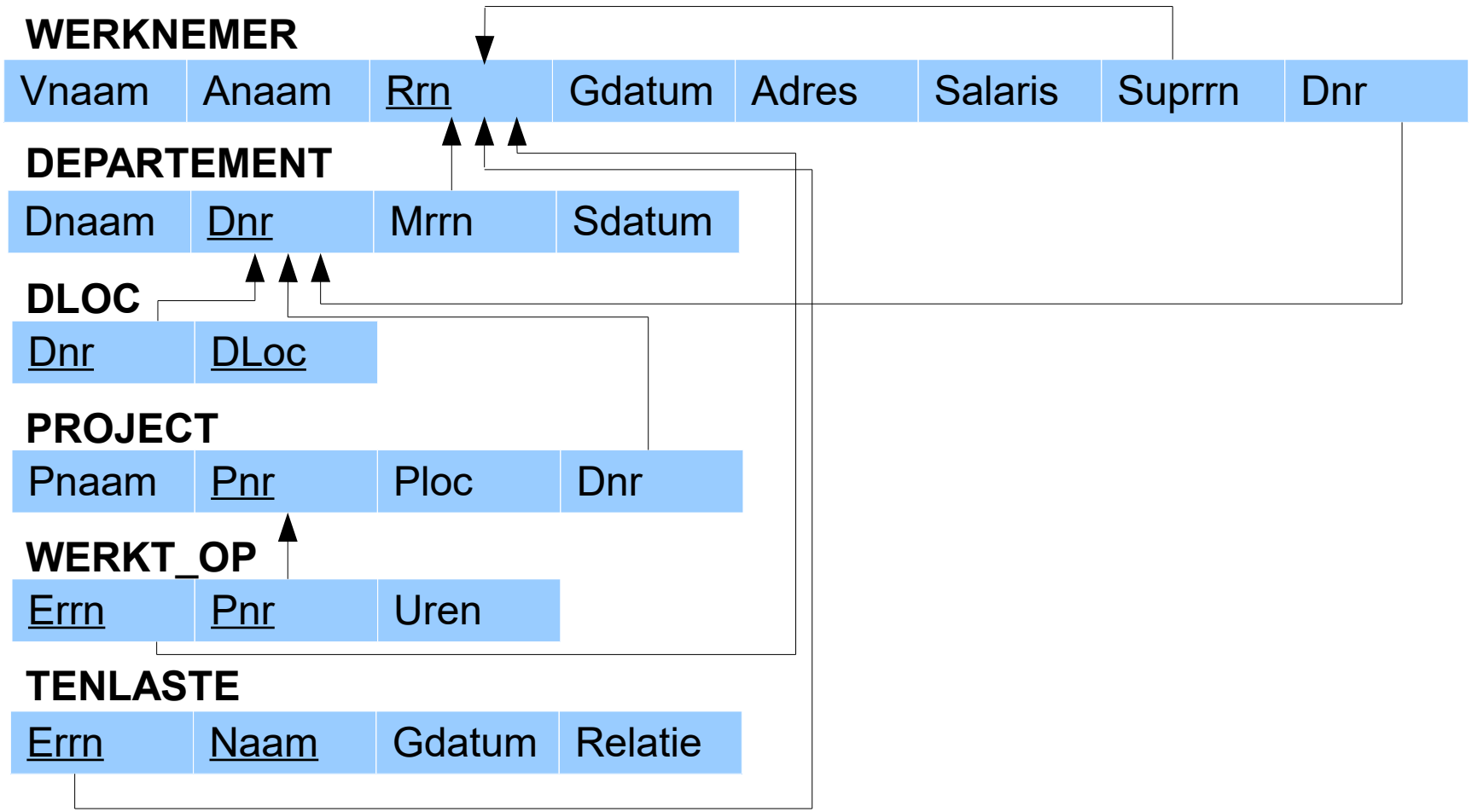
Voor elk meerwaardig attribuut A van relatie R

* Maak nieuwe relatie S in schema
* Met attribuut dat overeenkomt met A
* Verwijssleutel is de primaire sleutel K van bijhorende relatie R
* Primaire sleutel wordt A en K samen

DLOC: Dnr, DLoc



Doorlopend Voorbeeld: BEDRIJF



Stap 7 – Voor hogere-orde relatietypes

Voor elk hogere-order relatietype Q (graad>2)

* Maak nieuwe relatie S in schema
* Met verwijssleutel de primaire sleutels van deelnemende entiteitstypes
* Met primaire sleutel alle verwijssleutels samen
* Met enkelvoudige attributen van Q en samengestelde/meervoudige zoals eerder