ISO
Datamodelleren

Prof. dr. Paul De Bra

Gebaseerd op:
Database System Concepts, 5th Ed.

het bank voorbeeld

- waarom zijn er drie tabellen om klanten en rekeningen voor te stellen?

  customer (customer_name, customer_street, customer_city)

  account (account_number, branch_name, balance)

  depositor (customer_name, account_number)

- het proces dat tot deze keuze van tabellen en attributen leidt is *datamodelleren*

een database ontwerpen

- een database representeert de informatie van een bedrijf/organisatie
  - eerst en vooral: de informatie*behoefte* van de gebruikers vaststellen
  - dan de informatie modelleren in een *conceptueel model*
  - *functionele eisen* opstellen (welke *operaties* of *bewerkingen* op de informatie zijn nodig?)
  - het model *optimaliseren*: *redundantie* verwijderen en rekening houden met *incomplete informatie*

modelleren met het E-R model

- een *database instance* kan worden voorgesteld als:
  - een verzameling van *entiteiten*
  - verbanden (*relaties*) tussen die entiteiten
- een **entiteit** is een *object* dat bestaat en dat kan worden onderscheiden van andere objecten
  - voorbeeld: een bepaalde persoon, bedrijf, gebeurtenis, begrip
- entiteiten hebben *attributen*
  - voorbeeld: personen hebben *namen* en *adressen*
- een **entiteitenverzameling** is een verzameling entiteiten van hetzelfde *type* met dezelfde *eigenschappen*
  - voorbeeld: een verzameling personen, bedrijven, …

entiteitverzamelingen *customer* and *loan*

customer_id	customer_ name	customer_ street	customer_ city
321-12-3123	Jones	Main	Harrison
019-28-3746	Smith	North	Rye
677-89-9011	Hayes	Main	Harrison
555-55-5555	Jackson	Dupont	Woodside
244-66-8800	Curry	North	Rye
963-96-3963	Williams	Nassau	Princeton
335-57-7991	Adams	Spring	Pittsfield

customer

loan_ number	amount
L-17	1000
L-23	2000
L-15	1500
L-14	1500
L-19	500
L-11	900
L-16	1300

loan

verzamelingen van relaties (verbanden)

- een **relatie** is een verband tussen entiteiten
  voorbeeld:

  Hayes — *customer* entiteit
  depositor — relatie
  A-102 — *account* entiteit

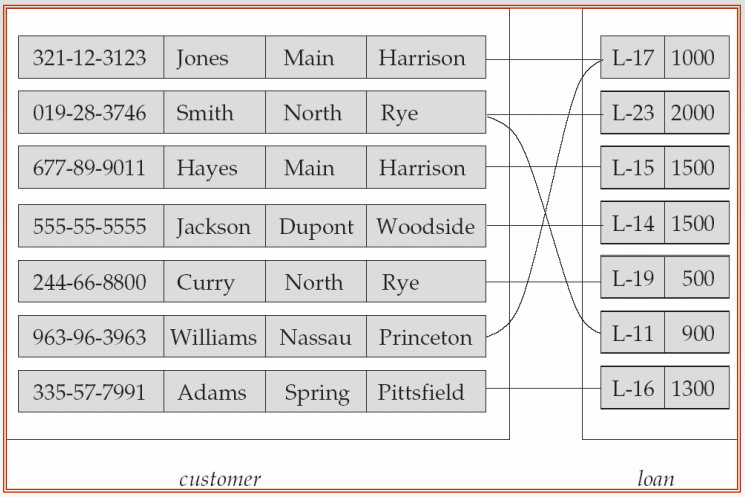
- een **relatie-verzameling** is een (wiskundige) relatie tussen $n \geq 2$ entiteit-verzamelingen

  $\{ (e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n \}$

  waarbij $(e_1, e_2, \ldots, e_n)$ een relatie (verband) is
  - voorbeeld:

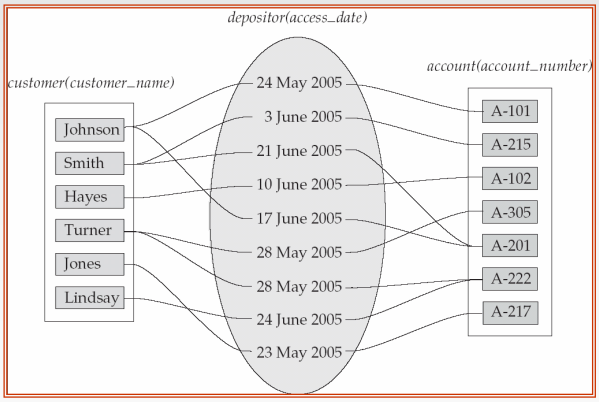
    (Hayes, A-102) $\in$ *depositor*

relatie-verzameling *borrower*



relatie-verzamelingen (cont.)

- een **attribuut** kan ook een eigenschap van een relatie-verzameling zijn.
- *depositor* verbindt niet alleen *customer* en *account* maar kan ook een attribuut *access-date* hebben



de “graad” van een relatie-verzameling

- de *graad* is het aantal entiteit-verzamelingen dat door de relatie-verzameling verbonden wordt
- de meeste relatie-verzamelingen zijn **binair** (van graad 2); denk aan *depositor* (verbindt *customer* met *account*) of *borrower* (verbindt *customer* met *loan*)
- relatie-verzamelingen van een hogere graad kunnen wel voor komen, maar wij gaan ze nooit gebruiken
  - Stel dat bedienden van een bank een functie kunnen hebben bij verschillende filialen, en bij verschillende filialen ook verschillende functies. Dan is er een *ternaire* relatie tussen *bediende, functie,* and *filiaal.*

attributen

- Een entiteit wordt *gerepresenteerd* door een verzameling attributen: ze beschrijven eigenschappen van alle elementen van een entiteit-verzameling

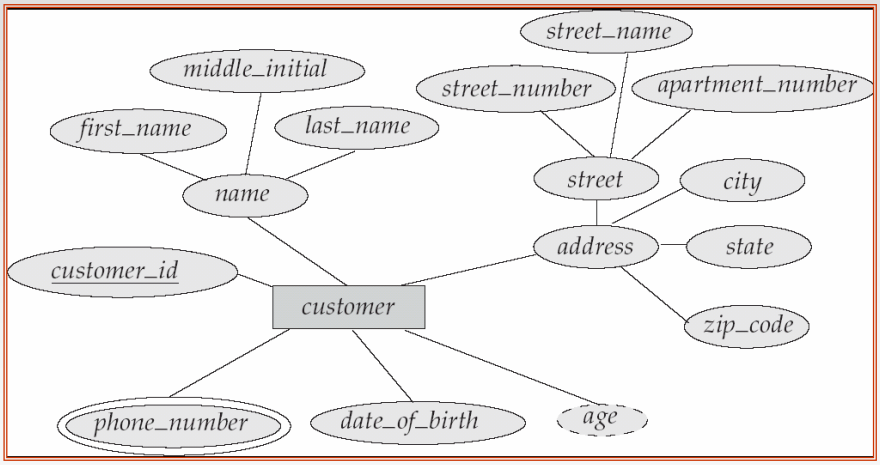
  voorbeeld: *customer* = (*customer_id, customer_name, customer_street, customer_city*)
  loan = (*loan_number, amount*)

- **Domain** (waardenverzameling) – de verzameling toegestane waarden voor elk attribuut
  - voorbeelden: voor *naam*: strings,
    voor *postcode*: 4 cijfers gevolgd door 2 letters
    voor telefoonnummer: 11 cijfers (vb: 31402472733)
    voor salaris: niet-negatief geheel getal

attributen (cont.)

- attribuut-types:
  - *enkelvoudige* en *samengestelde* attributen
    - een naam is: *voornaam* + *familienaam*
    - een adres: *straat* + *huisnummer* + *postcode* + *gemeente*
    - een postcode: *cijfercode* + *lettercode*
  - *een-waardige* en *meerwaardige* attributen
    - een-waardig: *naam* van een persoon
    - meerwaardig: *telefoonnummers* van een persoon
  - *afgeleide* attributen
    - niet afgeleid: *geboortedatum*
    - afgeleid: uit de *geboortedatum* leiden we de *leeftijd* af

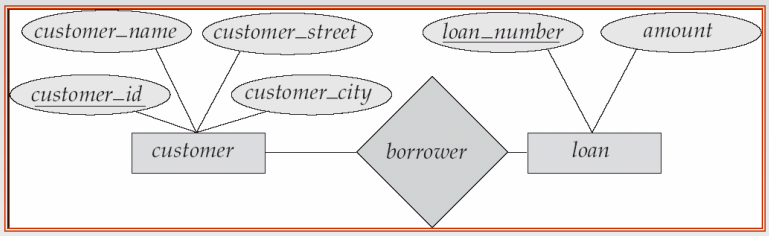
E-R diagram met samengestelde, meerwaardige, en afgeleide attributen



sleutels (herhaling)

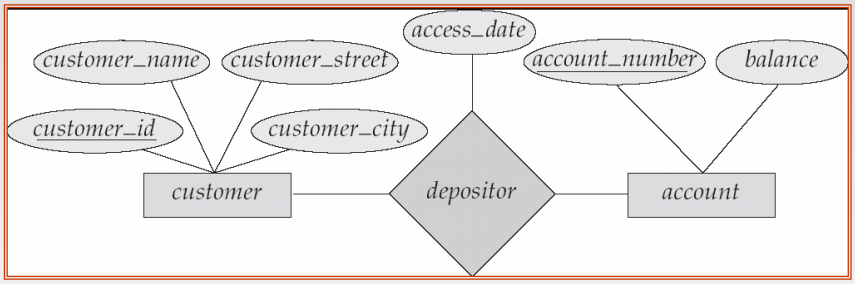
- een **supersleutel** van een entiteit-verzameling is een verzameling attributen waarvan de waarden (samen) de entiteit uniek identificeren
- een **kandidaat sleutel** voor een entiteit-verzameling is een minimale supersleutel
  - *customer_id* is kandidaat sleutel voor *customer*
  - *account_number* is kandidaat sleutel voor *account*
- de database ontwerper kiest uit de kandidaat sleutels een **primaire sleutel**
  - dit is meestal een kandidaat sleutel die uit slechts 1 attribuut bestaat (zoals *customer_id* en *account_number*) meer dat hoeft niet

E-R diagrammen



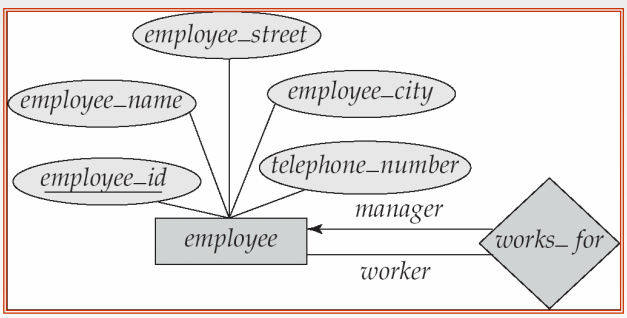
- entiteit-verzamelingen worden voorgesteld door rechthoeken
- relatie-verzamelingen worden voorgesteld door ruiten
- lijnen verbinden attributen met entiteit-verzamelingen of relatie-verzamelingen en verbinden entiteit-verzamelingen met relatie-verzamelingen
- attributen worden voorgesteld door ovalen (ellipsen)
  - een dubbele ovaal betekent een meerwaardig attribuut.
  - een gestippelde ovaal betekent een afgeleid attribuut
  - een onderstreepte attribuutnaam hoort bij een primaire sleutel

een relatie-verzameling met attributen



rollen

- een entiteit-verzameling mag een relatie hebben met zichzelf
  - we gebruiken labels, als *manager* and *worker* om de **rollen** aan te duiden: ze geven aan hoe de *employee* entiteiten interageren via de *works_for* relatieverzameling



sleutels van relatie-verzamelingen

- de combinatie van primaire sleutels van de betrokken entiteit-verzamelingen is een supersleutel voor de relatie-verzameling
  - (*customer_id, account_number*) is een supersleutel voor *depositor*
  - *OPGELET! dit betekent dat een paar entiteiten maar 1 keer kan voorkomen in een bepaalde relatie-verzameling*
- we moeten naar de cardinaliteit van de relatie-verzameling kijken om een kandidaat sleutel te kiezen
- we moeten de *betekenis* van de relatie-verzameling gebruiken om een *primaire sleutel* te kiezen (als er verschillende kandidaat sleutels zijn)

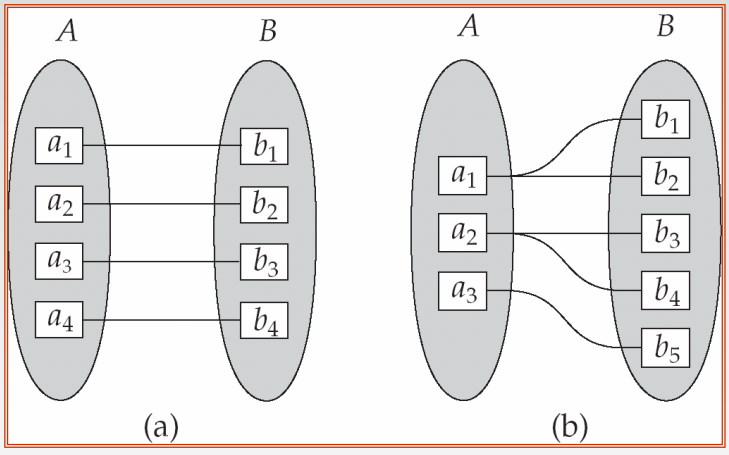
opgave (6.15)

- Maak een E-R diagram voor een ziekenhuis, met patienten en dokters.
  Het ziekenhuis moet een “log” bijhouden van de tests en onderzoeken die elke patient ondergaat.
- Merk op dat er meer dan 1 diagram (of datamodel) mogelijk is. Bespreek de voor- en nadelen van verschillende alternatieven.

beperkingen op een relatie-verzameling

- we kunnen het *aantal* verbindingen tussen entiteiten (via relaties) beperken
- **beperkingen** of **constraints** worden vooral gebruikt bij *binaire* relatie-verzamelingen
- we gebruiken 4 soorten binaire relaties:
  - een op een
  - een op veel
  - veel op een
  - veel op veel

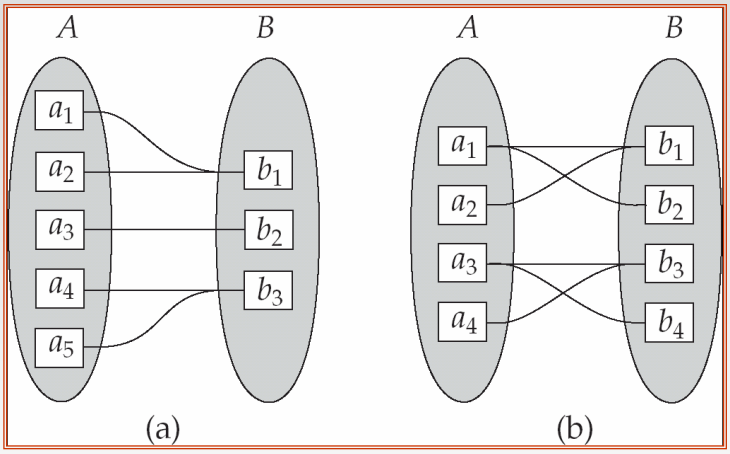
voorbeelden van "cardinaliteit"



een op een een op veel

het is mogelijk dat sommige elementen in *A* en/of *B* niet verbonden zijn met een element in de andere verzameling

voorbeelden van “cardinaliteit”



veel op een

veel op veel

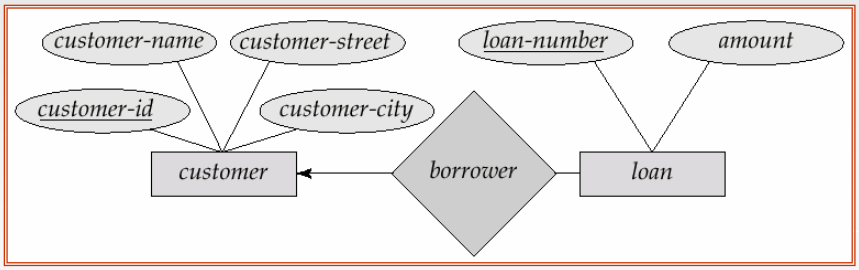
het is mogelijk dat sommige elementen in *A* en/of *B* niet verbonden zijn met een element in de andere verzameling

cardinaliteit-beperkingen (notatie 1)

- een pijl (→) betekent “één”, een lijn (—) betekent “veel”
- een-op-een verband (pijl naar beide kanten) bij *customer – borrower – loan* betekent dit:
  - een klant is verbonden met hoogstens 1 lening via de relatie *borrower*
  - een lening is verbonden met hoogstens 1 klant via de relatie *borrower*

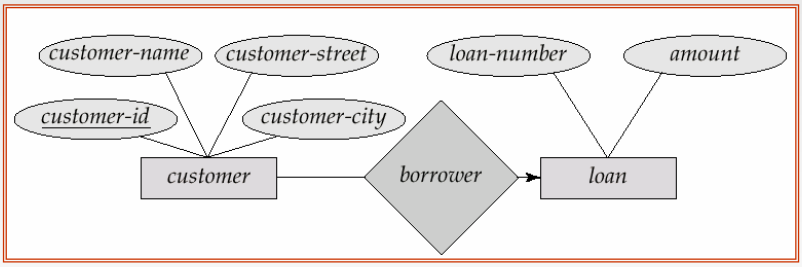
een-op-veel relaties

- een een-op-veel relatie bij *customer – borrower – loan* betekent dat elke lening via *borrower* verbonden is met hoogstens 1 klant, en dat een klant via *borrower* verbonden is met een aantal (mogelijk 0) leningen



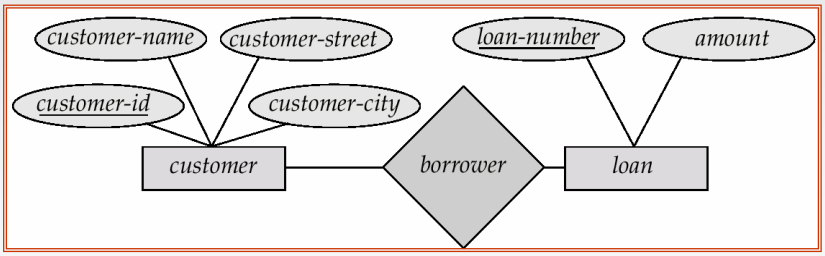
veel-op-een relaties

- een veel-op-een relatie bij *customer – borrower – loan* betekent dat elke lening via *borrower* verbonden is met een aantal (mogelijk 0) klanten, en dat een klant via *borrower* verbonden is met hoogstens 1 lening



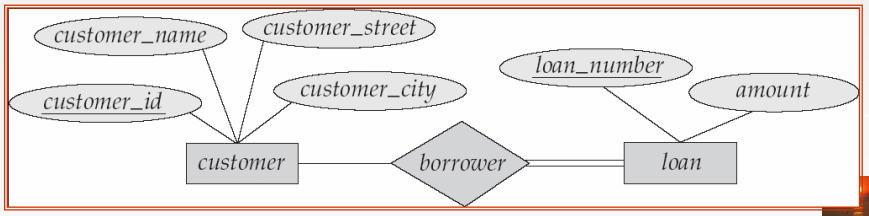
veel-op-veel relaties

- een klant is via *borrower* verbonden met een aantal (mogelijk 0) leningen
- een lening is via *borrower* verbonden met een aantal (mogelijk 0) klanten



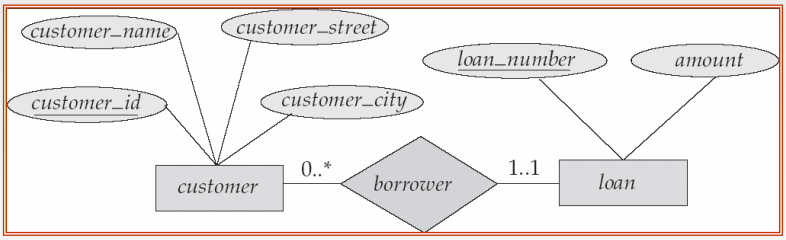
"verplichte" deelname in een relatie

- een dubbele lijn geeft een **totale** relatie aan: dus de aanduiding "mogelijk 0" uit de vorige sheets wordt dan "minstens 1".
  - vb: deelname van leningen in *borrower* is verplicht
    - dit betekent dat er bij elke lening minstens 1 klant hoort die de lening heeft afgesloten (en ermee verbonden is via *borrower*)
- partiele (niet totale) deelname: sommige entiteiten zijn niet via de relatie verbonden
  - vb: deelname van klanten aan de relatie *borrower* is niet verplicht
    - dit betekent dat niet elke klant minstens 1 lening moet hebben



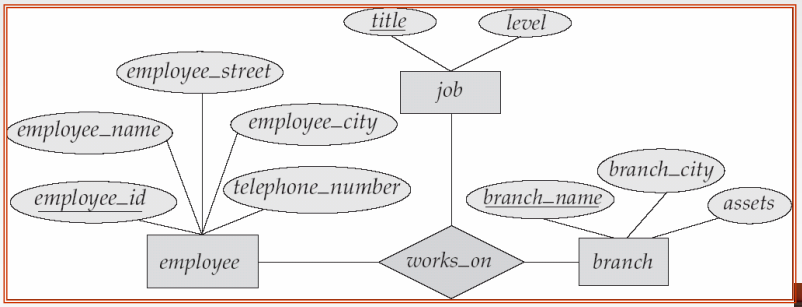
cardinaliteit-beperkingen (notatie 2)

- we kunnen getallen gebruiken om onder- en bovengrenzen aan te duiden
- dit is een krachtigere notatie: de ondergrens en bovengrens hoeven niet 0, 1 of veel (*) te zijn.
  - voorbeeld: een een-op-veel relatie *customer – borrower – loan* met verplichte deelname van *loan*:



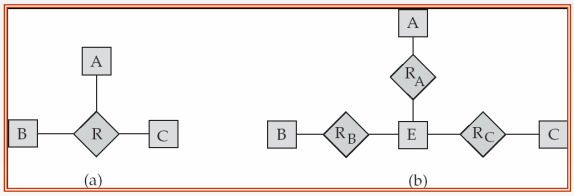
moeilijkheden met ternaire relaties

- wat betekent het als er een pijl staat bij *job*? (veel-op-een naar *job* toe)
- wat betekent het als er een pijl staat bij *job* en bij *branch*?



wegwerken van ternaire relaties

- vervang *R* (tussen A, B en C) door een entiteitverzameling *E*, en voeg drie relatieverzamelingen toe:
  1. R_A, tussen *E* en *A*
  2. R_B, tussen *E* en *B*
  3. R_C, tussen *E* en *C*
- geef *E desgewenst* een nieuw identificerend (sleutel) attribuut
- attributen van *R* worden attributen van *E*
- voor elke relatie (instance) (a_i , b_i , c_i) in *R,* creeer:
  1. een nieuwe entiteit e_i in *E*
  2. voeg (e_i , a_i) toe aan R_A
  3. voeg (e_i , b_i) toe aan R_B
  4. voeg (e_i , c_i) toe aan R_C

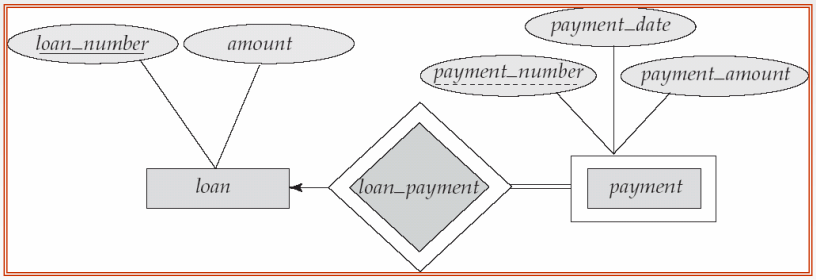


zwakke entiteitverzamelingen

- een entiteitverzameling zonder primaire sleutel is een **zwakke entiteitverzameling** (andere zijn **sterk**)
- een zwakke entiteit(verzameling) hangt af van een andere: de **identificerende entiteit(verzameling)**
  - er moet een totale een-op-veel relatie zijn van de identificerende entiteitverzameling naar de zwakke
  - we tekenen de identificerende relatieverzameling als een dubbele ruit
- de attributen die entiteiten van een zwakke entiteitverzameling onderscheiden vormen de **discriminator** (*of partiele sleutel*)
- de discriminator + de primaire sleutel = de primaire sleutel van de zwakke entiteitverzameling

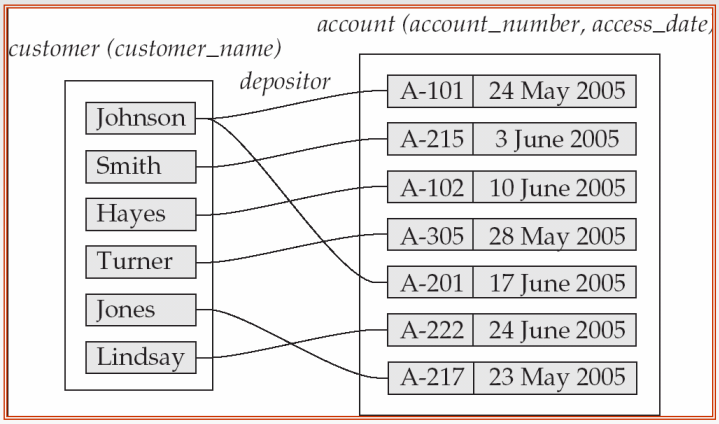
zwakke entiteitverzameling: voorbeeld

- zwakke entiteitverzameling: dubbele rechthoek
- discriminator gestippeld onderstreept
- *payment_number* – discriminator van *payment*
- primaire sleutel voor *payment* : (*loan_number, payment_number*)



cardinaliteit constraints beinvloeden het ontwerp

- stel er is een een-op-veel verband tussen *customer* en *account*. dan kunnen we een *access_date* toevoegen als attribuut bij *account* in plaats van bij de relatie *depositor.*



vertaling naar een relationeel schema

- De basis lijkt eenvoudig:
  - elke entiteitverzameling wordt een tabel
  - elke relatieverzameling wordt een tabel
  - attributen worden attributen
- Bij een eenvoudig schema is dit alles, maar:
  - ternaire relaties geven problemen (dus vermijden)
  - samengestelde attributen (moeten) verdwijnen
  - meerwaardige attributen vergen een speciale behandeling
  - er ontstaan “redundante tabellen” (te verwijderen)

vertaling entiteitenverzameling naar tabel

- een sterke entiteitverzameling wordt vertaald naar een tabel met dezelfde attributen
- een zwakke entiteitverzameling wordt vertaald naar een tabel met alle attributen van de entiteitverzameling + extra kolommen voor de primaire sleutel van de identificerende entiteitverz.

voorbeeld:
payment =

(*loan_number*, *payment_number*, *payment_date*, *payment_amount*)

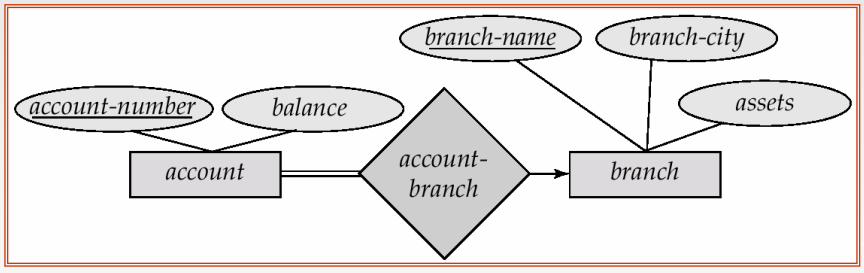
vertaling veel-op-veel relaties naar tabellen

- een veel-op-veel relatieverzameling wordt vertaald naar een tabel met:
  - attributen voor de primaire sleutels van de verbonden entiteitverzamelingen
  - elk attribuut van de relatieverzameling zelf wordt ook een attribuut van de tabel
- voorbeeld: neem de veel-op-veel relatie *depositor* met het extra attribuut *access_date*

  depositor = (*customer_id*, *account_number*, *access_date*)
- in principe kunnen ook een-op-een, een-op-veel en veel-op-een relaties zo vertaald worden, maar er zijn alternatieve oplossingen

veel-op-een en een-op-veel vertaling

- **totale** veel-op-een en een-op-veel relaties kunnen worden vertaald naar het toevoegen van de primaire sleutel van de “een” kant aan de tabel voor de entiteitverzameling aan de “veel” kant
- voorbeeld: in plaats van een *account_branch* tabel te maken voegen we een attribuut *branch_name* toe aan *account*



en de overige gevallen…

- een een-op-een verband kunnen we behandelen als een een-op-veel of veel-op-een verband
- als een relatieverzameling *partieel* is aan de “veel” kant dan kunnen we nog steeds attributen toevoegen aan de “veel” tabel (in plaats van een nieuwe tabel te maken), maar daarin kunnen dan waarden ontbreken
- voor de relatieverzameling die een zwakke entiteitverzameling verbindt met de identificerende entiteitverzameling hoeven we helemaal niets te doen
  - voorbeeld: de *payment* tabel bevat reeds alle attributen die in een *loan_payment* tabel zouden komen (i.e., *loan_number* en *payment_number*).

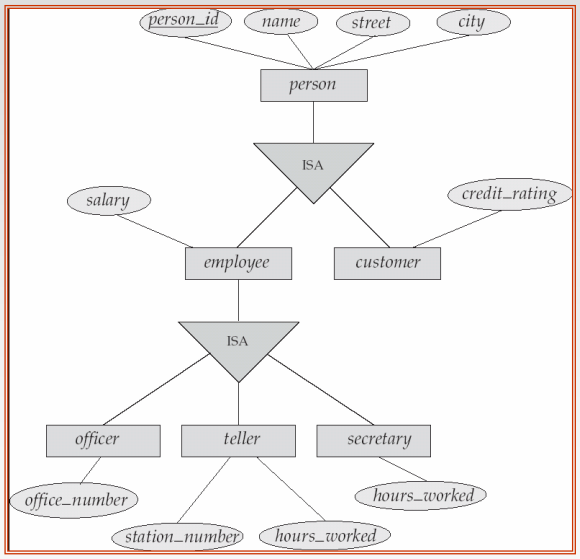
samengestelde en meerwaardige attr.

- samengestelde attributen worden “platgeslagen”
  - alleen de attributen van het laagste niveau blijven
- een meerwaardig attribuut *M* van een entiteitverzameling *E* vertalen we naar een tabel *EM*
  - *EM* heeft attributen voor de primaire sleutel van *E* en een attribuut *M*
  - elk element van een verzamelingswaardige *M* waarde genereert een tupel (rij) in tabel *EM*
    - voorbeeld: bij bedienden met een identiteitsnummer en een verzamelingswaardig “kinderen” attribuut, krijgen we voor bediende 123-45-6789 met kinderen Jack and Jane 2 tupels:
      (123-45-6789 , Jack) en (123-45-6789 , Jane)

specialisatie

- **komt uit top-down ontwerp**: we maken onderscheid tussen entiteiten in een entiteitverzameling
- de deel-groepen zijn entiteit-verzamelingen van een “lager niveau” en hebben extra attributen of nemen deel in relaties die niet van toepassing zijn op de hele (hoger niveau) entiteitverzameling
- we tekenen een ISA driehoek (vb. *customer* “is a” *person*).
- **Attribuut overerving** – een lager niveau entiteitverzameling erft alle attributen en deelname in relaties van de hoger-niveau entiteitverzameling (waarmee ze verbonden is)

voorbeeld van specialisatie



generalisatie

- **komt uit een bottom-up ontwerp proces** – een aantal entiteitverzamelingen met gemeenschappelijke eigenschappen (attributen of relatieverzamelingen) worden gecombineerd tot een hoger niveau entiteitverzameling
- gemeenschappelijke attributen en relaties worden “overgezet” naar de hoger niveau entiteitverzameling (en niet meer bij de lager niveau entiteitverzamelingen getekend)
- specialisatie en generalisatie zijn elkaars invers; in een E-R diagram zien ze er identiek uit; het is een verschil in modelleer-*proces*, niet in *resultaat*

specialisatie & generalisatie (cont.)

- je kan op basis van verschillende eigenschappen verschillende specialisaties definiëren
- vb: *permanent_employee* vs. *temporary_employee*, naast *officer* vs. *secretary* vs. *teller*
- elke medewerker kan in beide indelingen voorkomen:
  - hij is *permanent_employee* of *temporary_employee*,
  - en hij is ook *officer*, *secretary*, of *teller*
- de ISA relaties worden ook **superclass - subclass** relaties genoemd (de superclass is de hoger niveau entiteitverzameling, de subclass de lager niveau)

beperkingen (constraints) op specialisatie/generalisatie

- welke entiteiten mogen tot welke lager niveau entiteitverzameling behoren?
  - bepaald door een conditie, vb: bij *senior-citizen* ISA *person* horen alle personen ouder dan 65 tot de *senior-citizen* entiteitverzameling
  - gebruiker-bepaald, vb: wie is *customer*, wie is *employee*
- mogen entiteiten tot meer dan 1 lager niveau entiteitverzameling behoren?
  - **nee: disjoint**
    - we schrijven *disjoint* bij de ISA driehoek
  - **ja: overlapping**
    - we schrijven niets bij de ISA driehoek

beperkingen (constraints) op specialisatie/generalisatie (cont.)

- **volledigheid constraint** – geeft aan of elke entiteit in een hoger niveau entiteitverzameling een element moet zijn van een (of meer) lager niveau entiteitverzameling
  - **volledig (total)**: elke entiteit moet tot een lager niveau entiteitverzameling behoren
  - **partieel (partial)**: een entiteit hoeft niet tot een lager niveau entiteitverzameling behoren

vertaling van specialisatie naar tabellen

- methode 1:
  - een tabel voor de hoger niveau entiteitverzameling
  - een tabel voor elk van de lager niveau entiteitverzamelingen, met primaire sleutel van hoger niveau + alle “lokale” attributen

tabel	attributen
person	*name, street, city*
customer	*name, credit_rating*
employee	*name, salary*

  - nadeel: om alle informatie van een *employee* te tonen moet data uit twee tabellen gehaald worden

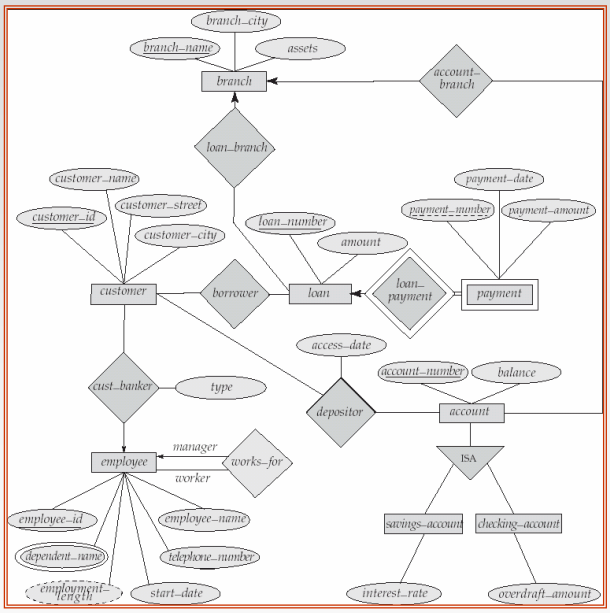
vertaling van specialisatie naar tabellen (cont)

- methode 2:
  - een tabel voor elke entiteitverzameling met alle “lokale” en alle ge-erfde attributen

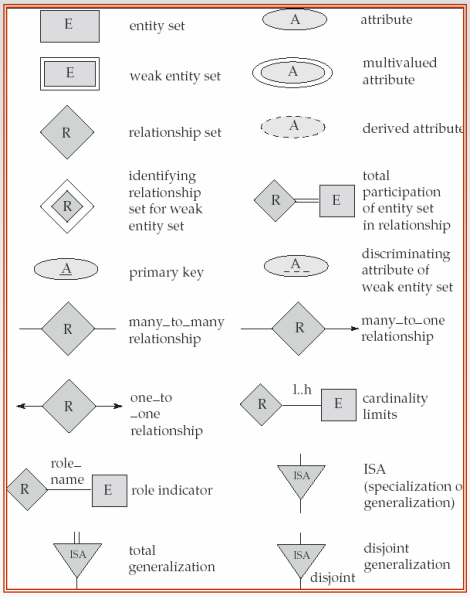
tabel	attributen
person	*name, street, city*
customer	*name, street, city, credit_rating*
employee	*name, street, city, salary*

  - als de specialisatie volledig (total) is dan is het schema voor de hoger niveau entiteitverzameling (*person*) niet nodig
  - nadeel: *street* en *city* wordt tweemaal opgeslagen voor personen die zowel *customer* als *employee* zijn

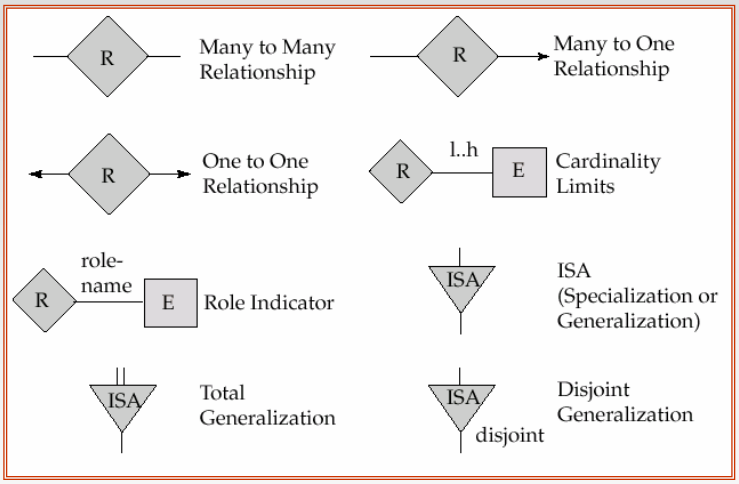
E-R diagram voor het bank voorbeeld



overzicht van alle symbolen uit E-R diagrammen



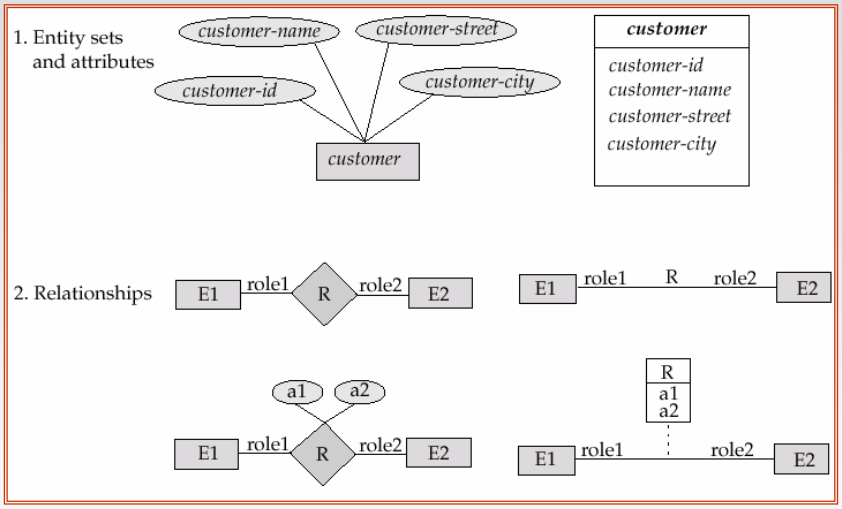
overzicht van symbolen (cont.)



UML

- **UML**: Unified Modeling Language
- UML bestaat uit verschillende onderdelen om verschillende aspecten van software systemen grafisch te modelleren
- UML klasse-diagrammen komen overeen met E-R diagrammen, maar ze gebruiken een verschillende teken-techniek (verschillende grafische syntax)

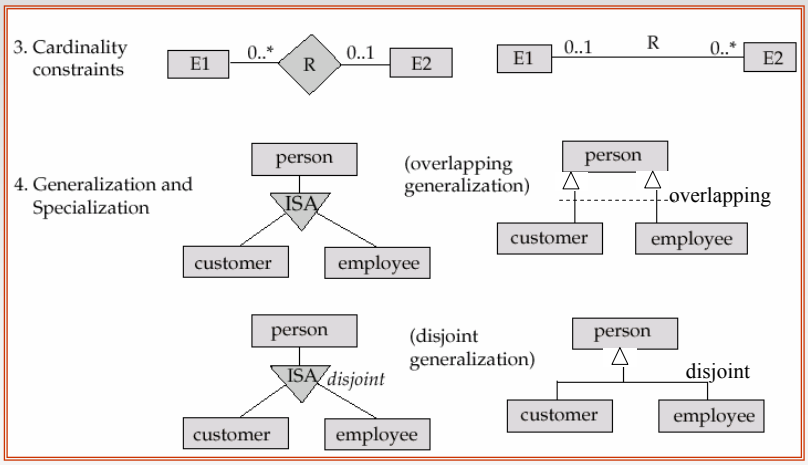
tekentechniek UML klasse diagrammen



UML klasse diagrammen (cont.)

- entiteitverzamelingen zijn rechthoeken, met attributen als een lijst binnen deze rechthoek
- binaire relaties worden voorgesteld door een lijn (zonder een ruit in het midden); de naam van de relatie staat neest deze lijn
- rollen kunnen ook bij deze lijn worden geschreven
- als een relatieverzameling attributen heeft wordt een rechthoek gemaakt met naam van de relatie en van de attributen
- voor niet-binaire relaties wordt een ruit gebruikt net zoals bij E-R diagrammen

UML klasse diagrammen (cont.)



*Opgelet: de plaats van cardinaliteitbeperkingen is omgewisseld

*Generalisatie kan met afzonderlijke pijlen of met een gemergde pijlen worden weergegeven

UML klasse diagrammen (cont.)

- cardinaliteitbeperkingen kunnen alleen in de vorm *l..h* worden weergegeven (niet met pijlen)
- opgelet de positie van de beperkingen is omgekeerd ten opzichte van E-R
- je mag 1 schrijven i.p.v. 1..1 en * i.p.v. 0..*