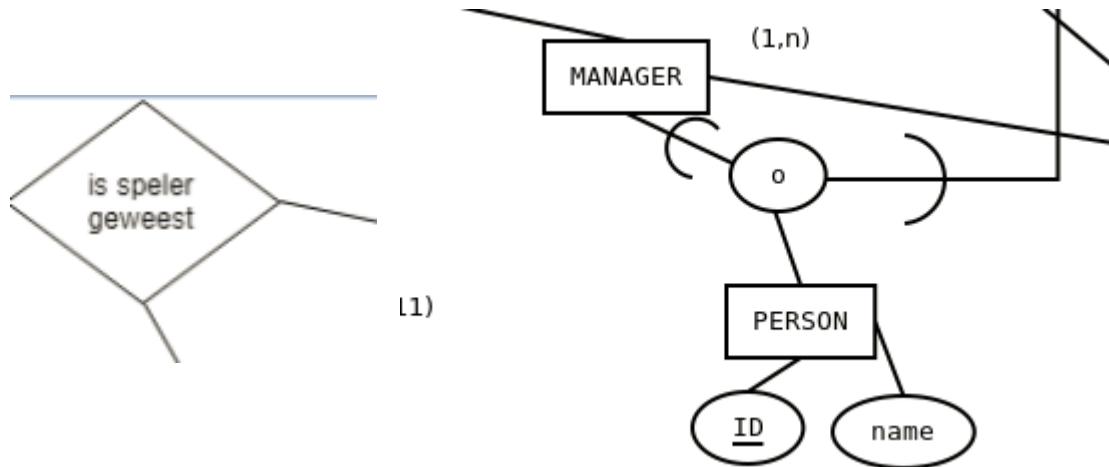


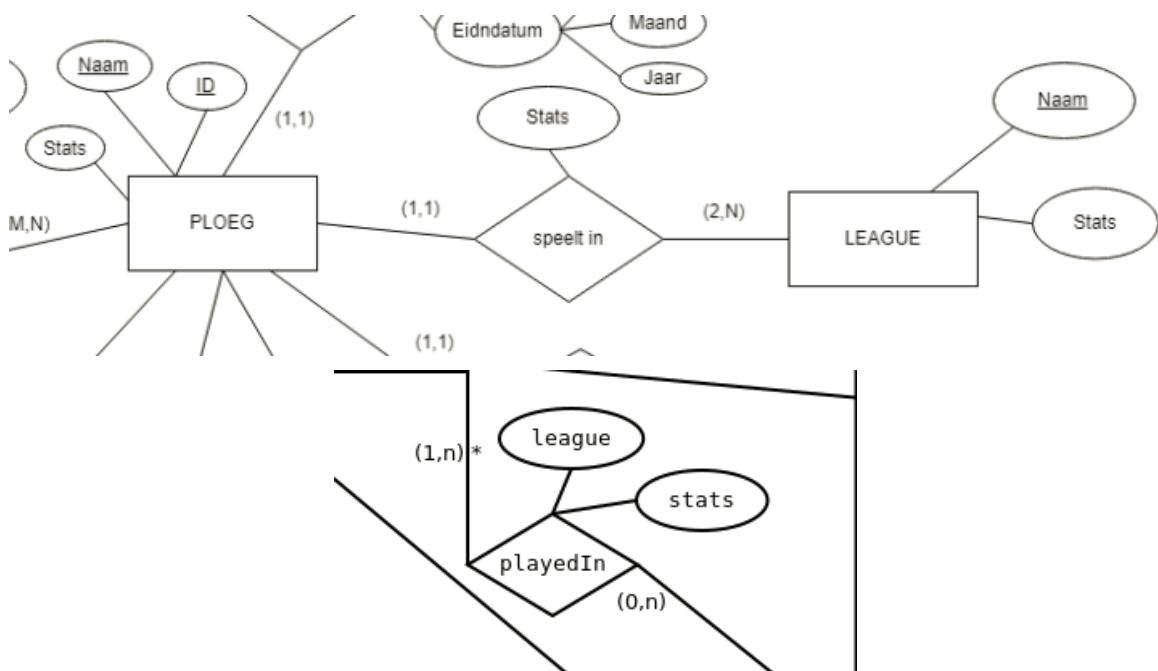
Gegevensbanken: Project Deel 2a

Verschil 1



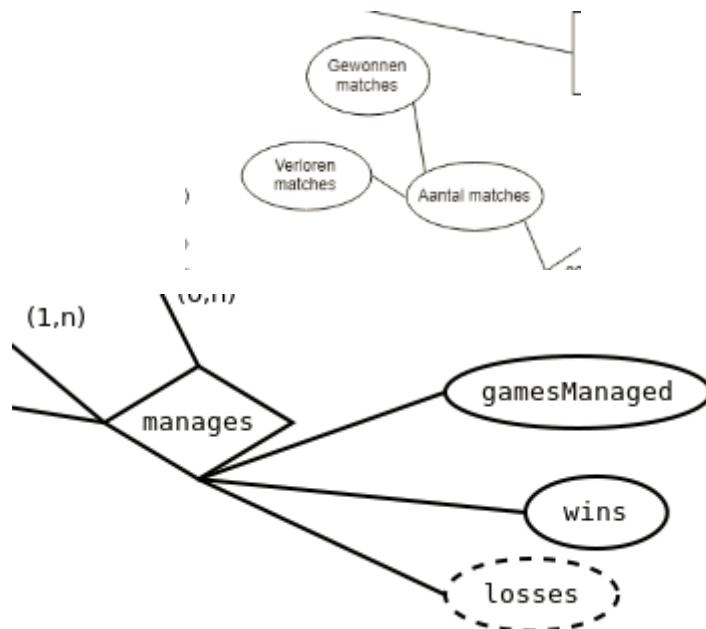
De oplossing beschrijft beter wat er in de opgave gesteld wordt. Het gaat hier namelijk over dezelfde persoon die een nieuwe functie opneemt en niet tussen een relatie tussen twee verschillende entiteiten. De gegevensbank voor de oplossing zal slechts één persoon bevatten, waar onze oplossing zowel een manager als een speler zal bevatten. Met als gevolg dat informatie twee maal wordt opgeslagen. De naam van een manager die speler is geweest is bijvoorbeeld gelijk voor de twee entiteiten en dient maar éénmaal opgeslagen te worden. Onze gegevensbank zal dus bepaalde informatie dubbel opslaan wat niet gunstig is voor het geheugen.

Verschil 2



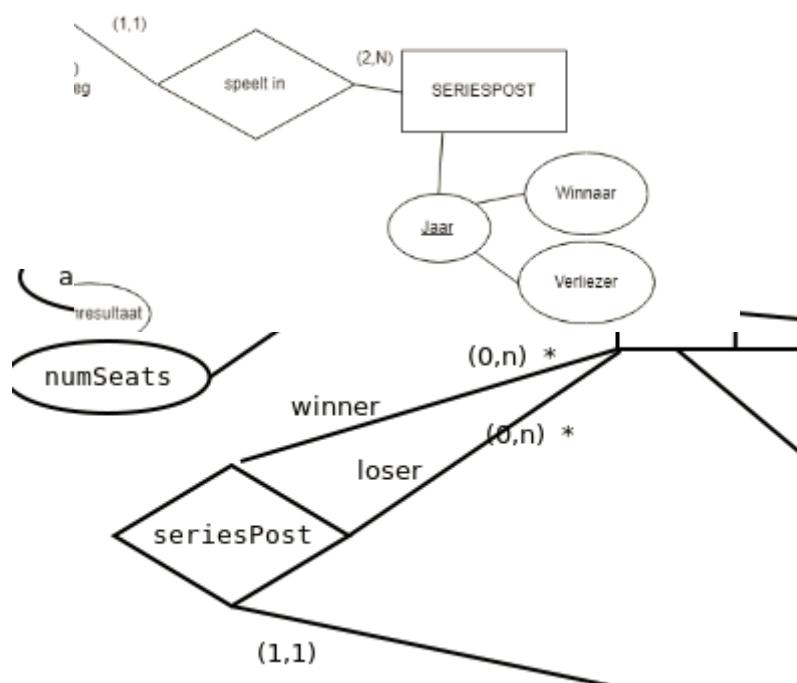
Wij hebben *league* als entiteit gemaakt. Dit zorgt ervoor dat in onze gegevensbank de ploegen die in een bepaalde league spelen samen gegroepeerd zijn in deze entiteit *league*. Bij de oplossing zal er een relatie *playedIn* aangemaakt worden met daarin de key attribute van zowel team en year. In onze gegevensbank zal er dus gezocht kunnen worden op *league*, terwijl dit voor de oplossing niet mogelijk is. Indien je deze functionaliteit nodig acht of niet is één van beide implementaties beter.

Verschil 3



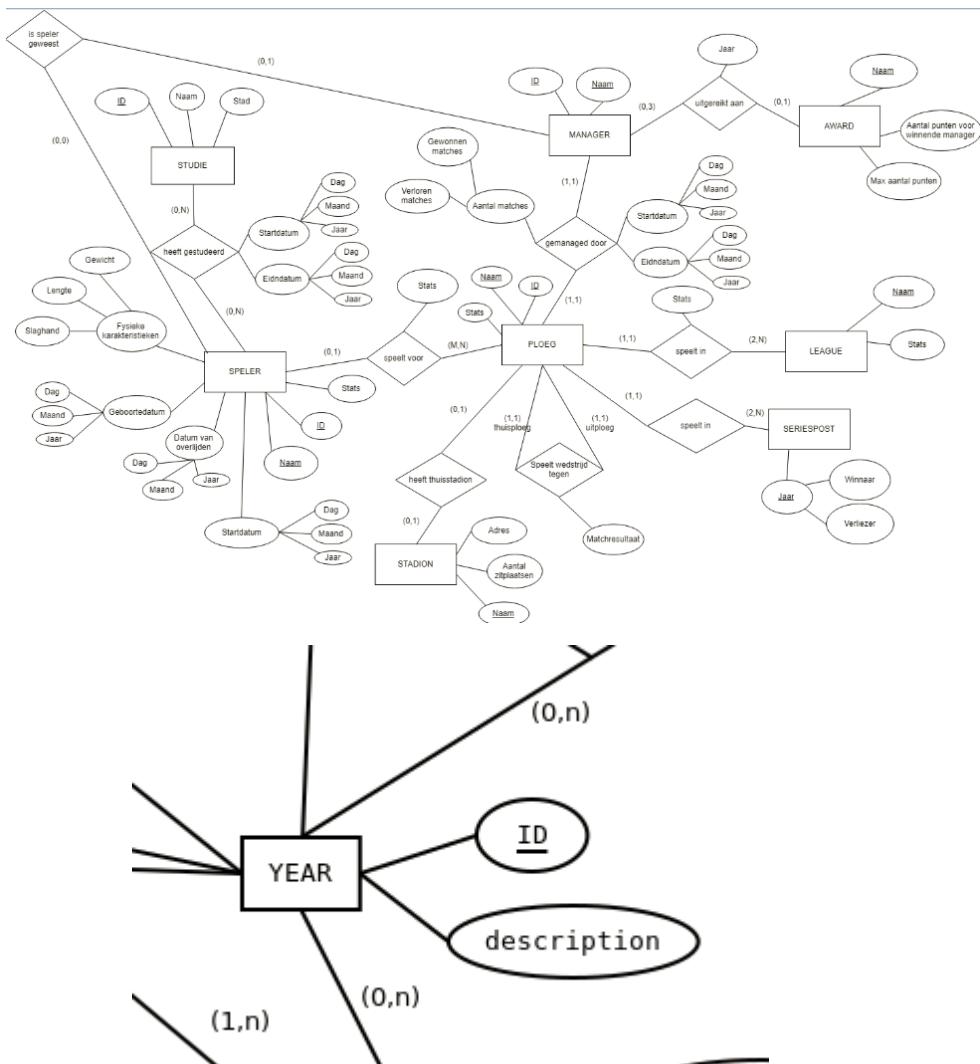
De oplossing maakt van de verloren matches een afgeleid attribuut. Dit is de betere keuze indien je de assumptie maakt dat er geen gelijkspel kan optreden, aangezien dit de redundantie in de tabel verminderd. Wij hebben een extra kolom met verloren matches in de tabel, terwijl deze afgeleid kan worden uit het totaal aantal matchen en de gewonnen matchen (wij hadden deze assumptie in het achterhoofd, maar zijn deze blijkbaar vergeten op te schrijven). Deze assumptie in achtig genomen hoort losses dus geen afgeleid attribuut te zijn. Dit is niet echt een zeer belangrijke fout, maar toont aan dat door simpele verbanden (gelijkspel toegelaten of niet) informatie niet hoeft opgenomen te worden in de gegevensbank. Wat geheugen spaart.

Verschil 4



In onze versie gebruiken we de min-max (1,1) tussen de relatie 'speelt in' en de entiteit 'Team'. Dit wordt verklaard in onze assumpties waar we aannemen dat elk team verplicht meedoet aan de seriespost. In de oplossing verwijst de min-max (0,n) naar het totaal aantal keer dat een team gewonnen of verloren is over de jaren heen. Een aparte entiteit seriespost laat toe om specifieke informatie over de seriespost op te slaan en op te vragen. Bij de ternaire relatie wordt echter in het relationeel model ook een relatie aangemaakt genaamd *SeriesPost* met key attributes van year en team waardoor de twee voorstellingen redelijk analoog zijn aan elkaar. Aangezien wij het jaar meegeven als attribuut aan de entiteit *SeriesPost*. Beide laten toe om analoge zaken op te vragen uit de *SeriesPost*.

Verschil 5



Waar in onze versie het jaar telkens als attribuut wordt meegegeven, wordt in de gegeven oplossing het jaar als entiteit geïmplementeerd. De tabel van de oplossing maakt het mogelijk om makkelijk gegevens per jaar op te zoeken, waar bij onze oplossing dit niet mogelijk is. Er kan enkel gecheckt worden of het jaar er tussen zit voor bijvoorbeeld een bepaalde ploeg, na opvraging van gegevens over deze ploeg. De implementatie van de opgave lijkt ons in dit geval dus de betere keuze. Opzoeken op jaar lijkt ons namelijk een handige functionaliteit. Indien je dit niet nodig acht, is onze implementatie een goed alternatief.

Hille Arnout, Soers Louis, Lagauw Sibren

Project Deel 2 (b)

Step 1: Mapping of Regular Entity Types.

+ Step 8D: Single relation with multiple type attributes.

PARK

Park_name, address, numSeats

TEAM

Team_name, team_ID

SCHOOL

School_ID, school_name, city

YEAR

Year_ID, description

PERSON (Step 8D)

Person_ID, person_name, playerFlag, birthdate, deathdate, weight, battingHand, height, debut, managerFlag

MGR AWARD

Award_Name

Step 2: Mapping of Weak Entity Types.

No weak entity types present.

Step 3: Mapping of Binary 1:1 Relationship Types.

PARK

Park_name, address, numSeats, team_ID

Step 4: Mapping of Binary 1:N Relationship Types.

→ We bekijken de relatie 'seriespost'
Als twee binaire relaties.

YEAR

Year_ID, description, team_ID

1:N(Team, Year) en M:N(Team, Team)

Step 5: Mapping of Binary M:N Relationship Types.

→ een andere optie is ze als ternair te beschouwen. Dan hoort de relatie seriespost bij stap 7.

SCHOOLSPLAYERS

School_ID, person_ID, yearMin, yearMax

ISACTIVEIN

Year_ID, person_ID, stats

PLAYEDIN

League, stats, team_ID, year_ID

SERIESPOST

Team_id

Step 6: Mapping of Multivalued Attributes.

No multivalued attributes present.

Step 7: Mapping of N-ary Relationship Types.

AWARDEDTO

Award_name, person_ID, year_ID, pointsWon, pointsMax

MANAGES

Person_ID, team_ID, year_ID, gamesManaged, wins

APPEARSIN

Person_ID, team_ID, year_ID, stats

Relational Database

PARK

Park_name, address, numSeats, team_ID

TEAM

Team_name, team_ID

SCHOOL

School_ID, school_name, city

YEAR

Year_ID, description, team_ID

PERSON

Person_ID, person_name, playerFlag, birthdate, deathdate, weight, battingHand, height, debut, managerFlag

MGRWARD

Award_Name

SCHOOLSPLAYERS

School_ID, person_ID, yearMin, yearMax

ISACTIVEIN

Year_ID, person_ID, stats

PLAYEDIN

League, stats, team_ID, year_ID

SERIESPOST

Team_id

AWARDEDTO

Award_name, person_ID, year_ID, PointsWon, PointsMax

MANAGES

Person_ID, team_ID, year_ID, gamesManaged, wins

APPEARSIN

Person_ID, team_ID, year_ID, stats